Hanna Kuhno

KETTERÄT MENETELMÄT JA CMMI: YHTEENSOPIVIA VAI -SOPIMATTOMIA?

Tietojärjestelmätieteen kandidaatintutkielma

5.2.2009

JYVÄSKYLÄN YLIOPISTO

TIETOJENKÄSITTELYTIETEIDEN LAITOS

TIIVISTELMÄ

Kuhno, Hanna Maria
Ketterät menetelmät ja CMMI: yhteensopivia vai -sopimattomia?/Hanna Kuhno
Jyväskylä: Jyväskylän yliopisto, 2009
47 s.
Kandidaatintutkielma

Tässä tutkielmassa tutustutaan ketterien menetelmien soveltamiseen CMMI (Capability Maturity Model Integration) –nimisen prosessien kypsyystasomallin yhteydessä. Tavoitteena on aihealueeseen tutustumisen lisäksi selvittää CMMI:n ja ketterien menetelmien yleisimmät yhteensopivuusongelmat sekä tuoda esille myös niihin kirjallisuudessa esitettyjä ratkaisuja. Niihin CMMI:n alueisiin, jotka ketterät menetelmät täyttävät hyvin, ei tutkielmassa puututa.

Ketterät menetelmät ja CMMI mielletään usein toistensa vastakohdiksi, joiden yhteensovittaminen on mahdotonta. Tutkielmassa aihetta käydään läpi esittelemällä ensin perusperiaatteet ketteristä menetelmistä sekä tutustumalla CMMI:n ja sen edeltäjän CMM:n (Capability Maturity Model) taustaan. Tämän jälkeen keskitytään tarkastelemaan suurimpia esteitä niiden yhdistämiselle.

Tämän kirjallisuuskatsauksen perusteella voidaan sanoa, että vielä aika ei ole kypsä näiden kahden lähestymistavan kitkattomalle yhteistyölle, mutta pyrkimystä yhteiseen ymmärrykseen on molemmin puolin havaittavissa.

AVAINSANAT: Ketterät menetelmät, Extreme Programming, Scrum, CMM, CMMI

Ohjaaja: Mauri Leppänen
Tietojenkäsittelytieteiden laitos
Jyväskylän yliopisto

Tarkastaja: Mauri Leppänen
Tietojenkäsittelytieteiden laitos
Jyväskylän yliopisto

SISÄLTÖ

1 JOHDANTO

Yhä useampi ohjelmistoja tuottava organisaatio hyödyntää ketteriä menetelmiä ohjelmistokehityksessään. Ohjelmistojen laajuuden ja monimutkaisuuden myötä myös ohjelmistoja tuottavien organisaatioiden kyvykkyys tuottaa laadukkaita tuotoksia on nostettu esille. Tässä tutkielmassa tarkastellaan yhden laadunvarmistusviitekehyksen, CMMI:n (lyhenne englannin kielen sanoista Capability Maturity Model Integration) (Software Engineering Institute, 2001), yhteensopivuutta ketterien menetelmien kanssa. Kuten tutkielman otsikostakin jo näkee, tästä yhteensopivuudesta ei olla kirjallisuudessa niinkään varmoja. Richard Turner ja Arpuva Jain (2002) aloittavatkin artikkelinsa osuvasti kuvaten ohjelmistoalan ammattilaisten suhtautumista tähän aiheeseen: ”Jos tyypilliseltä ohjelmistosuunnittelijalta kysyttäisiin, ovatko ohjelmistoille tarkoitettu kyvykkyyden kypsyystasomalli ja prosessin kehitys käyttökelpoisia myös ketterille menetelmille, vastaus todennäköisesti vaihtelisi tyhjästä tuijotuksesta hysteeriseen nauruun.”

Ketteristä menetelmistä puhuttaessa tarkoitetaan sellaisia ohjelmistokehityksen menetelmiä, joissa pyritään välttämään perinteisten tapojen (esimerkiksi vesiputousmallin) raskaina pidettyjä toimintamalleja (Boehm, 2002). Mutta mihin vedetään raja ketterän ja perinteisen menetelmän välillä? Abrahamsson ym. (2002) onnistuvat tiivistämään ketterien menetelmien oleellisimman sisällön seuraavasti: ”Tämä on se tapaus, kun ohjelmiston kehitys on vähittäisin muutoksin etenevää (pienten ohjelmistojen julkaiseminen nopeissa sykleissä), yhteistoiminnallista (asiakas ja ohjelmiston kehittäjät työskentelevät yhdessä ja kommunikoivat jatkuvasti), suoraviivaista (menetelmä itsessään on helppo oppia ja sitä on helppo muokata, hyvin dokumentoitu) ja mukautuvaa (mahdollisuus viime hetken muutoksiin)”

CMM (lyhenne sanoista Capability Maturity Model) (Paulk ym., 1993) ja CMMI (Software Engineering Institute, 2001) ovat taas enemmän perinteisiin menetelmiin soveltuvia, organisaation ja sen prosessien kypsyystason määrittelyyn suunniteltuja viitekehyksiä (Software Engineering Institute, 2007a). Niiden perimmäinen tarkoitus on tarjota mm. ohjelmistoalan organisaatioille joukko hyväksi havaittuja keinoja prosessien tehokkaampaan hallintaan (Software Engineering Institute, 2007a). CMMI:n avulla organisaatiolla on myös mahdollisuus arvioida ja kehittää eteenpäin omaa kypsyystasoaan hyödyntämällä siihen liittyvää arviointimenetelmää (Software Engineering Institute, 2007a). Kritiikkiä kirjallisuudessa CMMI on saanut aikaavievästä ja laajasta lähestymistavastaan (Pikkarainen, 2008).

Se syy, miksi ketterien menetelmien yhteensopimattomuus CMMI:n kanssa on herättänyt keskustelua alan asiantuntijoiden keskuudessa, johtuu siitä, että CMMI:hin liittyy myös virallisemman kypsyysluokituksen mahdollistava arviointimenetelmä, SCAMPI (lyhenne englannin kielen sanoista Standard CMMI Appraisal Method for Process Improvement) (Software Engineering Institute, 2006a, 2007a). Jos ketterin menetelmin ei ole mahdollista saavuttaa CMMI:n korkeampia kypsyystasoja, tarkoittaako se silloin sitä, että ketteriä menetelmiä hyödyntävä organisaatio ei ole yhtä hyvä kuin perinteisin menetelmin etenevä kilpailijansa?

Tämän tutkielman tarkoituksena onkin selvittää, miltä osin ja missä mielessä ketterät menetelmät tukevat prosessien kypsyystasoajattelua ja missä mielessä nämä ovat ristiriidassa keskenään. Sen selvittämiseksi tarkastellaan kirjallisuudessa esiintyviä ongelmia ja näihin ongelmiin ehdotettuja ratkaisuja. Tärkeimpiä lähteitä ovat sellaiset, joissa CMM:ää tai CMMI:tä verrataan johonkin tiettyyn ketterään menetelmään: Anderson (2005), Fritzsche ja Keil (2007), Kähkönen ja Abrahamsson (2004), Marcal ym. (2007), Paulk (2001), Pikkarainen ja

Mäntyniemi (2006) sekä Turner ja Jain (2002). Näiden lähteiden näkökulmia verrataan toisiinsa unohtamatta CMMI:n virallisia määritelmiä ja ketterien menetelmien manifestia. Käytännön hyötyä tästä tutkimuksesta on alalla työskenteleville ihmisille, jotka esimerkiksi harkitsevat CMMI:n hyödyntämistä ketteriä menetelmiä käyttävässä organisaatiossa. Tutkielma tarjoaa heille kuvauksen ongelma-alueista, joihin kannattaa erityisesti kiinnittää huomoita.

Tutkielman rakenne koostuu seuraavasti: luvuissa 2 ja 3 käydään läpi lyhyesti tutkielmassa käsiteltävien aiheiden taustaa ja niiden yleisiä toimintaperiaatteita. Luvussa 2 ketteristä menetelmistä nostetaan myös tarkempaan tarkasteluun Extreme Programming ja Scrum. Luvun 3 on tarkoitus johdattaa lukija CMM:n kautta CMMI:n rakenteeseen erityisesti ohjelmistotuotannon alueella. Luvussa 4 keskitytään tutkimusongelman mukaisesti tiettyjen ketterien menetelmien ja CMMI:n välisiin eroavaisuuksiin ja ristiriitoihin. Viimeisessä luvussa esitetään tutkielman tulokset ja johtopäätökset sekä pohditaan jatkotutkimuksen aiheita.

2 KETTERÄT MENETELMÄT

Tässä luvussa käydään läpi hyvin tiiviisti läpi ketterien menetelmien taustaa ja niiden yleiset periaatteet sekä syvennytään hieman paremmin kahteen ketteräksi luokiteltavaan menetelmään. Tarkoituksena on antaa lukijalle tiivistetty yleiskuva tämän hetken ohjelmistotuotannon ketteristä keinoista.

Sellaisia ohjelmistokehityksen tapoja, joissa pyritään välttämään perinteisten menetelmien (kuten esimerkiksi Winston Roycen esittelemän vesiputousmallin) raskaita toimintamalleja, kutsutaan *ketteriksi menetelmiksi* (Boehm, 2002). Menetelmät alkoivat kiinnostaa ohjelmistotuotannon ammattilaisia 1990-luvun loppupuolella, jolloin Kent Beck esitteli ensimmäisen kerran Extreme Programming – nimisen tavan kehittää ohjelmistoja (Abrahamsson ym., 2002). Kevyemmän ohjelmistokehityksen puolestapuhujia löytyi Beckin lisäksi muitakin. Heidän kokoonnuttuaan yhteen vuonna 2001 syntyi ”Ketterien menetelmien manifesti”, jossa ilmaistaan ketterien menetelmien perustavaa laatua olevat neljä arvoa (Beck ym., 2001):

- Yksilöitä ja vuorovaikutusta arvostetaan prosesseja ja työkaluja enemmän.
- Toimiva sovellus on tärkeämpi kuin kokonaisvaltainen dokumentaatio.
- Asiakasyhteistyö on tärkeämpää kuin sopimusten neuvottelut.
- Muutokseen reagoimista arvostetaan enemmän kuin suunnitelman noudattamista.

Näiden neljän arvon lisäksi ketterien menetelmien manifestiin kuuluu 12 yksityiskohtaisempaa periaatetta, joita tässä ei tarkemmin tarkastella (Beck ym., 2001). Ketterille menetelmille ei ole olemassa tämän tarkempaa virallista määritelmää, vaan yhteisenä tekijänä tähän joukkoon lukeutuvilla ohjelmistokehityksen menetelmillä on se, että ne noudattavat juuri esitellyn

ketterien menetelmien manifestin määrittelemiä arvoja ja periaatteita ohjelmistokehityksessään (Abrahamsson ym., 2002).

Kaksi useimmiten CMM:n tai CMMI:n kanssa vertailtua ketterää menetelmää ovat Extreme Programming ja Scrum. Molemmissa menetelmissä käsitellään ohjelmistotuotannon prosessia kuten perinteisessä vesiputousmallissa; analyysi, suunnittelu, toteutus ja testaus kuuluvat tiiviisti prosessin kiertokulkuun (Beck, 1999; Schwaber, 1995). Merkittävä ero ketterillä menetelmillä perinteisiin malleihin verrattuna on kuitenkin siinä, että nämä vesiputousmallista tutut vaiheet käydään läpi iteratiivisesti useaan kertaan ennen lopullista ohjelmiston valmistumista (Beck, 1999; Schwaber, 1995). Yhtäläisyyksistään huolimatta Scrumin ja Extreme Programming -menetelmän lähestymistavat poikkeavat kuitenkin osittain toisistaan, joten seuraavaksi käydään läpi kummankin toimintaperiaatteet lyhyesti.

2.1 Extreme Programming

Extreme Programming tunnetaan myös nimellä XP. Beck esitteli menetelmän perusperiaatteet artikkelissaan Embracing Change with Extreme Programming vuonna 1999. Monet XP:ssä olevat käytännöt olivat tuolloin jo olemassa, Beck vain esitteli nämä yhdessä paketissa. (Beck, 1999)

XP:ssä ohjelmistoprosessi jakautuu viiteen vaiheeseen, jotka ovat tutkimus, suunnittelu, iteraatio, tuotteistaminen sekä ylläpito ja kuolema.

- *Tutkimusvaiheessa* (engl. exploration phase) tutustutaan yleisesti kohdealueen toimintatapoihin sekä mietitään asiakkaan kanssa, mitä asioita halutaan ohjelmistoon toteuttaa (Abrahamsson ym., 2002).
- *Suunnitteluvaiheessa* (engl. planning phase) pohditaan asiakkaan kanssa, mitä toiminnallisuuksia halutaan toteuttaa ensimmäisessä julkaisussa (Abrahamsson

ym., 2002). Nämä toiminnallisuudet kirjoitetaan asiakkaan kanssa ylös niin kutsutuille tarinakorteille (engl. story cards) (Beck, 1999). Myös aikataulua arvioidaan alustavasti (Abrahamsson ym., 2002).

- *Iteraatiovaiheita* (engl. iterations to release) on useita. Ensimmäisissä iteraatioissa keskitytään toteuttamaan sellaisia tehtäviä, jotka tukevat järjestelmän rungon muodostumista. Asiakas päättää kunkin iteraation alussa, mitkä määritellyistä tehtävistä toteutetaan tulevassa iteraatiossa. Iteraation lopuksi suoritetaan asiakkaan määrittelemät toiminnalliset testit, joiden onnistuttua ohjelmisto julkaistaan. Jos toteutettavia tehtäviä on vielä julkaisun jälkeen tekemättä, jatketaan seuraavaan iteraatioon. Kukin iteraatio kestää yhdestä neljään viikkoon. (Abrahamsson ym., 2002)
- *Tuotantovaiheessa* (engl. productionizing phase) ohjelmistoa testataan entistä tarkemmin ja iteraatioiden kestoa saatetaan lyhentää viikkoon. (Abrahamsson ym., 2002)
- *Ylläpito- ja kuolemavaihe*: (engl. maintenance and death phase) ylläpitovaiheessa ohjelmisto on jo julkaistu asiakkaan käyttöön. Olemassaolevan ohjelmiston ylläpidon lisäksi projektin tehtäviin kuuluu tarvittaessa vielä uusien toiminnallisuuksien toteuttaminen. Kun asiakas on päättänyt, ettei lisää toiminnallisuuksia enää toteuteta, ollaan kuolemavaiheessa. Koska muutoksia ei enää tällöin tehdä, on loppudokumentaation vuoro. (Abrahamsson ym., 2002)

Edellä mainitut vaiheet ovat aika samankaltaisia seuraavaksi esiteltävän Scrumin kanssa, mutta XP:ssä on myös tiettyjä ominaispiirteitä, joita ei kaikissa ketterissä menetelmissä käytetä. XP hyödyntää pariohjelmoinnin tuomia etuja, virheiden minimoimiseksi kaikki kirjoitettu koodi on syntynyt kahden ohjelmoijan yhteistyönä (Paulk, 2001). Täten millään kirjoitetulla koodin osalla ei ole yksittäistä omistajaa, vaan kaikki saavat parannella ja muokata kirjoitettua koodia (Paulk, 2001). Projekteissa käytettävien tiimien koot on määritelty alle 10 ihmisen suuruisiksi, jolloin tiimin sisäinen kommunikointi on suhteellisen sujuvaa (Beck,

1999). Rajoittamalla työviikon tiukasti 40 tuntiin, toisin sanoen kieltämällä ylityötunnit, XP ottaa huomioon myös työntekijöiden jaksamisen (Vanderburg, 2005).

2.2 Scrum

Scrum terminä esiintyi kirjallisuudessa ensimmäisen kerran jo vuonna 1986, jolloin Hirotaka Takeuchi ja Ikujiro Nonaka käsittelivät artikkelissaan pienten tiimikokojen hyötyä tuotekehityksessä. Tästä ideasta hioutui myös ohjelmistotuotannon tarpeet huomioon ottava Scrum. Scrumin alkuperäinen perusperiaate on muodostaa sellaisia ohjelmistotuotannon tiimejä, joissa kaikki tiimin jäsenet olivat jatkuvasti tietoisia, missä yleisesti projektissa mennään. (Schwaber, 1995)

Menetelmän erikoinen nimi juontaa juurensa rugby-nimisen pelin aloitusmuodostelmasta (engl. scrum). Schwaber (1995) jaotteleekin artikkelissaan Scrumin kolme vaihetta pelin henkeen sopivalla tyylillä:

- *Esipeli* (engl. pregame) pitää sisällään aikataulutuksen ja hinnoittelun alustavan määrittelyn sekä järjestelmän arkkitehtuurin suunnittelun (Schwaber, 1995).
- *Peli* (engl. game) kuvaa Scrumin iteratiivista vaihetta, ns. sprinttivaihetta, jossa on itsessään neljä osaa: kehitysvaihe, kasausvaihe, tarkasteluvaihe sekä sopeutumisvaihe (Schwaber, 1995). Yksi erikoinen piirre Scrumin sprinttivaiheessa on se, että uuden sprintin alkaessa tiimillä on käsissään vain tuotteen tehtävälista (engl. product backlog) (Marcal, De Freitas, Furtado Soares, Marciel, & Belchior, 2008). Listan tehtävien muuntaminen toimivaksi ohjelmistoksi on täysin tiimin omalla vastuulla, jolloin työskentelytapana yritys

ja erehdyskään ei ole mitenkään kiellettyä (Marcal ym., 2008). Sprinttivaihe kestää XP:n iteraatiovaiheen tavoin viikosta neljään viikkoon (Schwaber, 1995).

- *Jälkipeli* (engl. postgame) pitää sisällään julkaisuun valmistautumisen vaiheet, loppudokumentaation tekemisen, testausta ennen julkaisua, sekä lopulta julkaisun (Schwaber, 1995).

Vaikka XP:ssäkin työskennellään tiimeissä, Scrumissa tiimit ovat hyvin olennainen osa menetelmää. Tiimit muodostuvat johtotiimistä ja kehitystiimistä (Schwaber, 1995). Johtotiimin tehtävänä on mm. valvoa projektin edistymistä ja pitää huolta aikataulussa pysymisestä (Schwaber, 1995). Kehitystiimiin kuuluu koodaajien lisäksi dokumentoija ja laadunvarmistaja (Schwaber, 1995). Tiimikoot Schwaber (1995) rajoittaa kolmesta kuuteen henkilöön, mutta tarvittaessa yhden projektin parissa voi työskennellä useitakin tiimejä (Schwaber, 1995). Tiimi kokoontuu yhteen joka päivä noin 15 minuutin ajaksi (Scrum-tapaaminen, engl. the daily scrum) ja tiimin jokainen jäsen vastaa omalta kohdaltaan seuraaviin kysymyksiin: mitä olet tehnyt viimeisen Scrum-tapaamisen jälkeen, mitä teet ennen seuraavaa tapaamista ja mitä vaikeuksia tai esteitä on ollut työssäsi (Marcal, de Freitas, Furtado Soares, & Belchior, 2007). Näiden tapaamisten lisäksi jokaisen sprinttivaiheen lopuksi pidetään erillinen tarkastelutapaaminen, jossa tiimin jäsenten ja projektin hallinnon lisäksi läsnä voi olla myös asiakkaita ja muita asianomaisia (Schwaber, 1995). Marcal ym. (2008) huomauttavat, että Scrum ottaa ohjelmistoprojektien hallintaan liittyvät asiat huomioon ohjelmointipainotteista XP:tä paremmin.

3 CMM JA CMMI

Tässä luvussa tutustutaan aluksi CMMI:n taustaan alkaen sen edeltäjästä, CMM:stä. CMM:n esittelyn yhteydessä käydään läpi myös sen alkuperäiset kypsyystasojen määritykset, jotka ovat pääpiirteittäin pysyneet samoina vielä tälläkin hetkellä käytössä olevissa versioissa. Tämän jälkeen keskitytään tarkastelemaan CMMI:n rakennetta erityisesti ohjelmistotuotannon näkökulmasta.

CMM (Capability Maturity Model) ja CMMI (Capability Maturity Model Integration) ovat molemmat prosessien kypsyystasomalleja. Näitä kypsyys-tasomalleja alettiin kehittää jo vuonna 1986, kun Software Engineering Institute (SEI) yhteistyössä Mitre Corp:in kanssa halusi rakentaa viitekehyksen auttamaan ohjelmistoalan kehittäjiä parantamaan ohjelmistoprosessejaan (Paulk, Curtis, Chrissis, & Weber, 1993). On tärkeää ymmärtää, että CMM tai CMMI ei ole itsessään prosessi, vaan malli, joka kuvailee tehokkaan prosessin ominaisuuksia (Software Engineering Institute, 2007a). Ensimmäinen maininta kehitteillä olevasta mallista oli Managing Software Process -nimisessä lehdessä 1987, mutta tarvittiin vielä neljä vuotta lisää, ennen kuin CMM:n versio 1.0 julkaistiin virallisesti (Paulk ym., 1993). Tämä ensimmäinen julkaisu CMM:stä oli kehitetty yleisesti ohjelmistotuotannon tarpeisiin, mutta myöhemmin mallista tehtiin eri versioita mm. järjestelmäkehityksen, ohjelmistotekniikan sekä työvoiman hallinnan ja kehityksen näkökulmat huomioon ottaen (Software Engineering Institute, 2001). Käytännön hyötyä CMM:n tai CMMI:n käyttöönotosta on esimerkiksi organisaatioille, joiden menestyminen markkinoilla vaatii jonkinlaista näyttöä toiminnan kypsyydestä (Fritzsche & Keil, 2007).

3.1 CMM

CMM:n kehitykselle lähtökohtina ovat toimineet yleiset ohjelmistokehityksen alalla olevat ongelmat (Paulk ym., 1993):

- aikataulujen ja budjettien ylittyminen
- yhteisesti hyväksyttyjen toimintamallien puuttuminen
- vastuualueiden ja roolien epäselvyys organisaatiossa

Näiden ongelmien pohjalta CMM:ssä on määritelty viisi eri tasoa organisaation prosessien kypsyydelle (engl. maturity levels) sekä tasot yksittäisten prosessien kyvykkyydelle (engl. capability levels). Seuraavaksi on esitelty CMM:n version 1.1 mukaiset organisaation prosessien kypsyystasot (Paulk ym., 1993):

- Taso 1: lähtötaso (engl. initial). Lähtötasolla organisaation menestyminen on täysin riippuvainen siellä työskentelevistä ihmisistä ja heidän tietotaidostaan. Projektit jäävät usein sovitusta aikataulusta ja budjetista jälkeen, koska organisaation tasolla syitä tähän ei analysoida eikä niiden ratkaisemiseksi ole kehitetty ohjeistusta.
- Taso 2: toistettavissa oleva (engl. repeatable). Tasolla 2 organisaatiossa on määritelty yhteiset pelisäännöt, joiden noudattamista projekteissa vaaditaan ja valvotaan. Hyväksi havaitut toimintamallit on poimittu aikaisempien kokemusten perusteella samankaltaisista projekteista. Projektien onnistumiset ovat toistettavissa.
- Taso 3: määritelty (engl. defined). Määritellyllä tasolla organisaatio on standardoinut prosessinsa sekä hallinnon, että ohjelmistosuunnittelun osalta. Henkilökunnalle annetaan johdonmukaisesti koulutusta, jotta tarvittava tietotaito ja yleisesti käytössä olevat toimintamallit siirtyisivät tasapuolisesti

kaikille. ”Prosessin kyvykkyys perustuu yleiseen, organisaation laajuiseen ymmärrykseen käytettävistä toiminnoista, rooleista ja vastuista määritellyssä prosessissa.”

- Taso 4: johdettu (engl. managed). Edellisiin tasoihin verrattuna tasolla 4 suurin muutos on siinä, että tässä vaiheessa mukaan tulee määrällinen laadunvarmistus. Prosesseista ja niiden tuottavuudesta ja laadusta kerätään konkreettista dataa, jonka avulla myöhemmin voidaan ennustaa mm. tulevien projektien mahdolliset riskialueet. Eri variaatiot prosessien suorituskyvyssä voidaan erotella toisistaan ja vaihtelun ollessa liian suurta asiaan osataan myös puuttua.
- Taso 5: optimoiva (engl. optimizing). Tasolla 5 jatketaan edellisellä tasolla kerätyn datan tulkintaa vielä pidemmälle. Tällä tasolla prosessien suorituskykyä pyritään parantamaan jatkuvasti tunnistamalla niissä esiintyvät heikkoudet, analysoimalla ne ja täten ehkäisemällä tunnettujen vikatyyppien syntymisen tulevissa projekteissa. Prosessien tehokkuus pyritään myös hiomaan huippuunsa tutkimalla niiden kulujen suhdetta saatavaan hyötyyn. Tasolla 5 puhutaan ensimmäisen kerran kroonisesta jätteestä (engl. chronic waste), jota muodostuu Paulkin (1993) mukaan suoritettavien tehtävien uudelleentyöstämisestä. Kroonista jätettä pyritään vähentämään taso tasolta, mutta päästäkseen tasolle 5 organisaatio ei saa tuottaa kroonista jätettä lainkaan. Jätteeseen johtavia syitä on alati analysoitava ja tehottomuuteen johtaviin syihin puututtava välittömästi.

3.2 CMMI

Tasojen määritelmät ja niiden nimet ovat hieman muuttuneet CMM:n kehittyessä kohti CMMI:tä, mutta perusidea kypsyystasoissa on pysynyt samana. CMMI:stä on kirjoitushetkellä kolme eri alalle erikoistunutta versiota:

- CMMI ohjelmistokehitykselle (engl. CMMI for development (CMMI-DEV)), jonka versio 1.02 on julkaistu jo vuonna 2000. Nykyinen versio on 1.2, joka on julkaistu vuonna 2006. Nimensä mukaisesti tämä malli on kehitetty erityisesti ohjelmistokehityksen tarpeet huomioon ottaen. (Software Engineering Institute, 2006b)
- CMMI palveluille, versio 1.2 (engl. CMMI for services (CMMI-SVC)) on vielä kehitysvaiheessa oleva malli, joka tarjoaa ohjeistusta palveluiden tarjoajille. Malli julkaistaneen vuoden 2009 puolella. (Software Engineering Institute, 2008b)
- CMMI hankinnalle, versio 1.2 (engl. CMMI for Acquisition (CMMI-ACQ)) on julkaistu marraskuussa 2007 ja sen pääpaino on tuotteiden ja palvelujen hankintaan liittyvissä toiminnoissa. (Software Engineering Institute, 2007b)

Koska tutkielman aihe keskittyy ohjelmistotuotantoon, tästä lähtien CMMI:llä tarkoitetaan nimenomaan ohjelmistokehitykselle suunnattua CMMI-DEV:n versiota 1.2, ellei toisin mainita.

CMMI:n ohjeistusta voidaan organisaatiossa toteuttaa kahdella eri tavalla. Ensimmäinen tapa on käydä vähitellen läpi peräkkäisiä prosessialueryhmiä asteittaisessa (engl. staged) kehittymisessä (Software Engineering Institute, 2006b). Tällöin puhutaan koko organisaation kypsyystasoista, jotka esiteltiin hieman laajemmin jo CMM:n yhteydessä (Software Engineering Institute, 2006b). Toinen tapa on jatkuva (engl. continuous) kehittyminen, jossa organisaatio saa vapaasti päättää ne prosessialueet, joissa koetaan tarvittavan parannusta (Software Engineering Institute, 2006b). Tämän tavan yhteydessä puhutaan jo aikaisemmin mainituista yksittäisten prosessien kyvykkyystasoista (Software Engineering Institute, 2006b). Mutta koska tutkielman aihepiiri on kytköksissä nimenomaan asteittaisen kehityksen kypsyystasomalleihin, näihin yksittäisten prosessien kyvykkyystasoihin ei jatkossa paneuduta.

CMMI:ssä on kullekin kypsyystasolle tietyt prosessialueet, jotka on asteittaisessa lähestymistavassa saavutettava ennen seuraavalle tasolle siirtymistä (Software Engineering Institute, 2006b). Jokainen prosessialue itsessään sisältää tarkemmin määriteltyjä erityistavoitteita (engl. specific goals) sekä yleistavoitteita (engl. generic goals) (Software Engineering Institute, 2006b). Prosessialueet voidaan sisältönsä perusteella jakaa neljään eri kategoriaan: prosessin hallintaan (engl. process management), projektijohtamiseen (engl. project management), suunnitteluun (engl. engineering) ja tukiprosesseihin (engl. support) (Software Engineering Institute, 2006b). Koska kypsyystasojen prosessialueita on CMMI:ssä 22 kappaletta, ne on suomennettu erityistavoitteineen erikseen liitteessä 1. Kaikkia prosessialueita koskevat yleistavoitteet on suomennettu liitteessä 2. Erityistavoitteet ja yleiset tavoitteet ovat sellaisia, jotka organisaation täytyy täyttää suorittaakseen kyseisiin tavoitteisiin liittyvän prosessialueen ja laajemmalti kyseisen kypsyystason hyväksytysti (Kähkönen & Abrahamsson, 2004). Kunkin erityistavoitteen ja yleistavoitteen jälkeen liitteissä on mainittu myös niihin liittyvät erityiskäytännöt tai yleiset käytännöt. Näitä käytäntöjä ei ole tarkoitus seurata yhtä orjallisesti kuin tavoitteita, ne ovat ennemminkin tapoja, joilla tavoitteet yleensä saavutetaan organisaatioissa (Kähkönen & Abrahamsson, 2004).

Kunkin kypsyystason ja niihin liittyvien prosessialueiden täyttäminen arvioidaan käyttäen CMMI-malleihin perustuvaa SCAMPI-menetelmää (lyhenne tulee englannin kielen sanoista Standard CMMI Appraisal Method for Process Improvement) (Kähkönen & Abrahamsson, 2004). SCAMPI:ssa on kolme eri tasoa, A, B ja C, joista ensimmäinen on ainoa virallisempaan luokitukseen oikeuttava taso (Software Engineering Institute, 2006a). Tutkielman pääpaino on kuitenkin CMMI:n tasoittaisessa esityksessä, joten SCAMPIn laajempi käsittely ei ole perusteltua.

4 KETTERIEN MENETELMIEN JA CMMI:N VÄLISET YHTEENSOPIVUUSONGELMAT

Tässä luvussa käsiteltävät CMMI:n ja ketterien menetelmien väliset yhteensopivuusongelmat on ehkä helpoin käydä läpi CMMI:n kypsyystasojen avulla. Luvun runko noudatteleekin Fritzschen ja Keilin (2007) artikkelia, jossa kunkin kypsyystason prosessialueet on käyty läpi taso kerrallaan. Kaikkia CMMI:n prosessialueita ei tuoda kypsyystasojen kanssa esille, vaan käsittelyyn otetaan vain sellaisia prosessialueita, joiden yhteensovitus ketterien menetelmien kanssa on todettu kirjallisuudessa vaativaksi tai mahdottomaksi. Yleistavoitteita ei ole käyty erikseen läpi, koska niissä esiintyvät ongelmat heijastuvat eri prosessialueisiin. Koska kaikkien organisaatioiden uskotaan olevan vähintään tasolla 1, sitä ei tässä luvussa käsitellä lainkaan. Ennen varsinaista yhteensopivuusongelmien esittelyä käydään pintapuolisesti läpi vertailussa käytettävien lähteiden tutkimus-menetelmät.

4.1 Käytettävien lähteiden tutkimusmenetelmät

Yhteensopivuusongelmien selvittäminen tehdään vertailemalla ja esittelemällä kirjallisuudessa esitetyt argumentit aiheesta. Pääasiallisia lähteitä on seitsemän kappaletta, joiden tutkimusmenetelmiä käydään läpi seuraavaksi hieman tarkemmin.

Andersonin (2005) artikkelissa huomion arvoista on muista lähteistä selkeästi poikkeava ketterä menetelmä. Hän viittaa W. Edwards Demingin oppeihin, joita ei kyseisessä artikkelissa käydä kuitenkaan kokonaisuudessaan läpi. Kirjoittaja pitää näitä oppeja ketterän menetelmän mukaisina viitaten seitsemään kohtaan Demingin esittelemässä johtamistavassa, joissa nähdään eniten yhtymäkohtia

ketterien menetelmien manifestin kanssa. Tällaisia Andersonin mainitsemia yhtymäkohtia ovat mm. työssäoppiminen, johtajuus, luottamuksen luominen asiakkaiden kanssa sekä laadukkaaseen tulokseen keskittyminen. Näiden oppien pohjalta Anderson selvittää artikkelissaan oman ratkaisunsa CMMI:n ja Demingin ketterien oppien yhdistämiselle ainakin tasolle 3 asti, eikä tason 5 saavuttaminenkaan ole hänen johtopäätöksensä mukaan ongelmallista. (Anderson, 2005)

Muissa artikkeleissa käsitelty ketterä menetelmä on yleisimmin joko Scrum tai XP, mutta muutama muukin menetelmä on mukana. Kähkönen ja Abrahamsson (2004) käytti empiirisessä kokeessaan laajennettua XP:tä. Suurin osa lisätyistä toiminnoista oli lähes suoraan lainattuja Scrumista, joiden lisäksi otettiin käyttöön mm. tiedonkeruuta helpottavia toimintoja, ylimääräistä dokumentaatiota sekä lisättiin kokoonpanon hallinnan toimintoja (Kähkönen & Abrahamsson, 2004). Empiirinen kokeilu suoritettiin oikeassa kahdeksan viikon mittaisessa projektissa, jossa oli mukana neljä ohjelmoijaa (Kähkönen & Abrahamsson, 2004). Muista käsiteltävistä artikkeleista poiketen tämän projektin päätyttyä siitä suoriutuminen arvosteltiin virallisella SCAMPI-asteikolla, jonka tuloksena oli tason 2 luokitus (Kähkönen & Abrahamsson, 2004).

Pikkaraisen ja Mäntyniemen (2006) artikkelissa taas esiintyy useita eri ketteriä menetelmiä, joita vertailtiin muutamaan valikoituun CMMI:n prosessialueeseen erityistavoitteiden tasolla. Heidän tutkimuksensa tulokset koostettiin yhteensä seitsemästä eri projektista, pääosin haastattelujen ja työpajojen (engl. workshop) perusteella (Pikkarainen & Mäntyniemi, 2006). Myös Turner ja Jain (2002) ovat käyttäneet hyväkseen työpajaa tietojenkeruutapana. Heidän tutkimuksessaan käsitellään mielipiteitä, jotka on kerätty vuonna 2002 pidetyn ketterien menetelmien työpajan yhteydessä (Turner & Jain, 2002).

Fritzsche ja Keil (2007), Marcal ym. (2007) sekä Paulk (2001) ovat kaikki ottaneet teoreettisemman lähestymistavan tutkimuksissaan. Kaikissa kolmessa artikkelissa kirjoittajat ovat pohtineet aihetta käymällä läpi valitsemiaan prosessialueita yksi kerrallaan ja perustelemalla, miten hyvin tai huonosti heidän käsittelemänsä ketterä menetelmä täyttää prosessialueen erityistavoitteet (Fritzsche & Keil, 2007; Marcal ym., 2007; Paulk, 2001). Tutkielmassa on pyritty ottamaan huomioon eri CMM:n ja CMMI:n versiot, etenkin Paulkin tapauksessa, esittelemällä vain sellaisia argumentteja ja mielipiteitä, jotka ovat edelleen samassa linjassa CMMI-DEV:n kanssa. Taulukossa 1 eritellään tiivistetysti käytettyjen lähteiden lähestymistavat aiheeseen.

TAULUKKO 1: käytettävien lähteiden tutkimusmenetelmät

Kirjoittajat	Julkai-su-vuosi	Käsitelty kypsyystasomalli	Käsitelty ketterä menetelmä	Menetelmä	Käsittelyalue
Anderson	2005	CMMI-SE/SW/IPPD/SS, V1.1, asteittainen malli 2002	W. Edwards Demingin ketterä menetelmä	Teoreettinen analysointi	Lähinnä taso 3
Fritzsche & Keil	2007	CMMI-SE/SW/IPPD/SS, V1.1	XP	Teoreettinen analysointi	Kaikki tasot
Kähkönen & Abrahamsson	2004	Nokia CMMI-Based Process Assessment (Standard CMMI Appraisal Method for Process Improvement (SCAMPI), V1.1)	Laajennettu XP	Empiirinen kokeilu yhdessä projektissa	Tason 2 prosessialueet, paitsi toimittajasopimusten hallinta
Marcal ym.	2007	CMMI-DEV, V1.2, asteittainen malli	Scrum	Teoreettinen analysointi	Projektijohtamisen prosessialueet
Paulk	2001	SW-CMM	XP	Teoreettinen analysointi	Kaikki SW-CMM:n avainprosessialueet
Pikkarainen & Mäntyniemi	2006	CMMI-SE/SW/IPPD/SS, V1.1, asteittainen malli 2002	Scrum, Scrumin ja XP:n yhdistelmä, Mobile-D	Empiirinen tutkimus: 3 eri organisaatiota, 7 eri projektia	Projektinhallinnan, vaatimusten hallinnan ja tasojen 2-3 suunnittelun prosessialueet
Turner & Jain	2002	CMMI-SE/SW/IPPD, V1.1, jatkuva malli 2001	Ketterät menetelmät	Kyselytutkimus	Prosessialueet ja yleiset tavoitteet

4.2 Taso 2: toistettavissa oleva

Toimittajasopimusten hallinta on prosessialue, jonka toteuttaminen ketterissä menetelmissä voi olla Fritzschen ja Keilin (2007) mukaan ongelmallista. Jos ketterään projektiin tarvitaan ulkopuolisilta toimittajilta komponentteja, niiden myöhästyminen iteraation lopussa voi tarkoittaa, että toimivaa ohjelmistoa ei saada koottua (Fritzsche & Keil, 2007). Mutta sekä Paulk (2001) että Turner ja Jain (2002) toteavat kuitenkin ongelman olevan hyvin harvinainen, kun otetaan huomioon, että ulkopuolisilta toimittajilta tilaaminen ei ole kovin yleistä ketteriä menetelmiä käytettäessä.

Projektin suunnittelun osalta Marcal ym. (2007) huomauttavat ketterissä menetelmissä puutteen työn tuotoksen ja tehtävien ominaisuuksien arvioinnissa. Esimerkiksi XP:n tarinakorttien sisältöä tai Scrumin työlistan tehtäviä ei ole rajattu millään tavalla kokonsa tai monimutkaisuutensa osalta (Marcal ym., 2007). Myös datan hallinnan suunnittelussa ja tarvittavan tietotaidon ja kykyjen määrittelyssä on Marcalin ym. (2007) mukaan puutteita. Näihin kohtiin ei kuitenkaan ole muussa kirjallisuudessa puututtu ongelmallisina, koska ketterien menetelmien nähdään täyttävän alueet riittävän hyvin keräämällä dataa lyhyen aikavälin suunnitelmista esimerkiksi tarinakorttien tai työlistan muodossa (Fritzsche & Keil, 2007). Projektissa tarvittavien tietotaidon ja kykyjen määrittely voidaan ottaa huomioon esimerkiksi lisäämällä ketteriin menetelmiin niistä muodollisempi suunnitelma. (Kähkönen & Abrahamsson, 2004). Vaikka ketteriä menetelmiä käytettäessä harvoin syntyy konkreettista näyttöä projektin suunnittelusta, Pikkarainen ja Mäntyniemi (2006) huomauttavat suunnittelun olevan oleellinen osa iteraatioita. Esimerkiksi Scrumin tapauksessa tuotteiden tehtävälistaa (engl. product backlog) voidaan pitää kirjallisena dokumenttina projektin suunnittelusta (Pikkarainen & Mäntyniemi, 2006).

Projektin valvonta ja hallinta on useimpien tutkimuksien mukaan hallittavissa myös ketterin keinoin, mutta Marcal ym. (2007) tarttuvat edelleen datan hallintaan ja eritoten datan hallinnan riittävään tarkkailuun. Turner ja Jain (2002) ilmaisevat myös mielipiteiden vaihtelun asiassa riippuen tulkinnasta, miten paljon esimerkiksi datan arkistoimista tai sen julkaisua pitäisi suunnitella, mutta ei silti nosta asiaa suurimpien ongelmien joukkoon (Software Engineering Institute, 2006b; Turner & Jain, 2002). Tässä tapauksessa mielipiteiden eroaminen johtuneekin CMMI:n tulkitsemisesta hyvin eri tavoin. CMMI hakee näillä kerättävään dataan liittyvillä erityistavoitteilla periaatteessa vain sitä, että turhaa dokumentaatiota tulisi välttää. Kun asiaa tarkastelee tältä kantilta, ei ketteriä menetelmiä tästä voi juurikaan syyttää, vaikka varsinaisia suunnitelmia tarvittavan datan keruusta ei sen paremmin olisikaan.

Mittaus ja analyysi on alue, jossa Turner ja Jain (2002) ovat ainoat muutamia soraääniä esiin nostaneet kirjoittajat. Osa heidän tutkimukseensa osallistuneista ihmisistä olivat pitäneet mittausta ja analyysiä itsessään ketterien menetelmien ulkopuolisena toimintona, ristiriitaisena prosessialuetta ei kuitenkaan ketterien menetelmien kanssa pidetty (Turner & Jain, 2002).

4.3 Taso 3: määritelty

Organisaation prosessien määrittely aiheuttaa keskustelua muutamassa artikkelissa. Fritzsche ja Keil (2007) tulkitsevat, että ketterissä menetelmissä tähän prosessialueeseen ei puututa lainkaan. Ilmeisesti tämän mielipiteen takana on oletus, että ketterien menetelmien ei ole tarkoituskaan puuttua organisaation tasolla tapahtuvaan prosessien määrittelyyn. Turnerin ja Jainin (2002) kyselytutkimuksessa osallistujien mielipiteet vaihtelivat merkittävästi riippuen siitä, nähdäänkö ketterä menetelmä itsessään hyväksi havaittujen prosessien tietolähteeksi vai ei.

Prosessit organisaation näkökulmasta ovat ristiriidassa ketterän menetelmätavan kanssa, koska ketterissä menetelmissä keskitytään nimenomaan meneillä olevaan projektiin, eikä oteta kantaa prosessien hallintaan organisaation kannalta (Fritzsche & Keil, 2007). Ketterissä menetelmissä ongelmat ratkotaan samalla kun niitä ilmenee, eikä näin syntyvää tietämystä ongelmanratkaisuista dokumentoida projektin ulkopuolisten käyttöön mitenkään (Paulk, 2001). Myös Turner ja Jain (2002) toteavat tutkimuksessaan tämän selkeäksi konfliktiksi CMMI:n kanssa edellä mainituista syistä (Turner & Jain, 2002).

Suurimmat syyt ketterien menetelmien kehittämiseen olivat juuri aikataulun venymiseen, muuttuviin vaatimuksiin ja näihin asioihin liittyviin riskitekijöihin liittyvät ongelmat. Tältä kantilta katsottaessa *riskienhallinnan prosessialueen* tavoitteet on hyvin katettu (Fritzsche & Keil, 2007). Mutta eriäviäkin mielipiteitä löytyy. Riskienhallinnan katsotaan tällöin ulottuvan ketterissä menetelmissä vain lyhyen aikavälin ongelmiin, jolloin pitkän aikavälin riskienhallinnan puute koetaan olevan ristiriidassa CMMI:n kanssa (Turner & Jain, 2002; Marcal ym., 2007).

Päätöksenteon analyysi on selkeimmin ristiriidassa ketterien menetelmien perusperiaatteiden kanssa. Tämä prosessialue on tarkoitettu kaikkien muiden prosessialueiden tukitoimeksi helpottamaan päätöksentekoa tarjoamalla siihen hyvin virallisen arviointiprosessimallin (Software Engineering Institute, 2006b). Tiukkaan päätöksentekoprosessiin tukeutuminen jokaisen eteen tulevan ongelman kohdalla ei ole linjassa ketterien menetelmien manifestin viimeisen kohdan kanssa, jossa alleviivataan muutokseen sopeutumisen tärkeyttä (Turner & Jain, 2002).

4.4 Taso 4: johdettu

Organisaation prosessien suorituskykyä pyritään mittaamaan luomalla sille tiettyjä tavoitteita ja lähtökohtia, joiden toteutumista tarkkailemalla saadaan myös mittauksellista dataa (Software Engineering Institute, 2006b). Turner ja Jain (2002) nostavat esille tämän CMMI:n prosessialueen ristiriitaisuuden ketterien menetelmien kanssa: jäykkien toimintamallien noudattaminen ei anna sijaa vastata jatkuviin muutoksiin, eikä se myöskään tue keskittymistä yksilöihin ennemmin kuin prosesseihin ketterän manifestin mukaisella tavalla (Fritzsche & Keil, 2007; Turner & Jain, 2002). Heidän tutkimuksessaan nousi esille myös toisenlainen näkökulma, jossa tulkittiin esimerkiksi Scrumissa olevan sellaisia mittauksellisia elementtejä, jotka täyttäisivät tämän prosessialueen vaatimukset (Turner & Jain, 2002).

Myöskään *määrällinen projektin johtaminen* ei ole samoilla linjoilla ketterien menetelmien kanssa (Marcal ym., 2007). Fritzsche ja Keil (2007) nostavat esille kaksi perustelua tälle väitteelle. Ensinnäkin he huomauttavat, että tilastollisissa menetelmissä tarvitaan välttämättä jokin pysyvä mitta-asteikko, johon esimerkiksi prosessin suoritusta voidaan verrata (Fritzsche & Keil, 2007). Toinen argumentti väitteen takana on se, että useimmiten ketterät menetelmät hyödyntävät pieniä tiimikokoja, jolloin tehtyjen mittausten tulokset eivät välttämättä ole tilastollisesti käyttökelpoisia (Fritzsche & Keil, 2007).

4.5 Taso 5: optimoiva

Organisaation innovaatio ja järjestäytyminen aiheutti Turnerin ja Jainin (2002) tutkimuksessa kahden eri ääripään mielipiteen muodostumisen. Toinen puolisko oli sitä mieltä, että tämä prosessialue kuvaa hyvin ketterien menetelmien

toimintatapoja (Turner & Jain, 2002). Tämän mielipiteen kannattajat näkevät asian ilmeisesti siltä kantilta, että esimerkiksi prosessialueen ensimmäinen erityistavoite, ”valitse parannukset”, kuvaa nimenomaan sitä toimintaa, mitä mm. Scrumissa tapahtuu sprintin aikana. Sprintissä vastaantulevat ongelmat ratkotaan, miten parhaaksi nähdään ja hyväksi havaittuja keinoja hyödynnetään myös myöhemmissä vaiheissa. Eriävän mielipiteen edustajat perustelevat ristiriidan mm. sillä faktalla, että mitään CMMI:n määrittelemiä käyttöönotettujen parannusten mittauksia ketterissä menetelmissä ei tehdä (Turner & Jain, 2002). CMMI:n erityistavoitteita vapaammin tulkitseva Anderson (2005) näkee kuitenkin mahdollisuuden myös tilastolliseen tutkintaan. Hänen mielestään esimerkiksi mittaamalla asiakkaan valitsemien toimintojen toteuttamiseen kulunutta aikaa, saadaan CMMI:n määrittelemää tilastollista dataa tulevaisuuden suunnittelua varten (Anderson, 2005). Fritzsche ja Keil (2002) tarttuvat kuitenkin siihen, että ketterissä menetelmissä parannukset ja uudet innovaatiot ovat vain projektien sisäisiä asioita, organisaation tasolta katseltuna prosessialueen erityistavoitteet eivät tällöin toteudu.

Syysuhteen analyysi ja ratkaiseminen nähdään ketteriin menetelmiin verratessa neutraalina prosessialueena (Turner & Jain, 2002). Ketterät menetelmät eivät toteuta prosessialueen ominaistavoitteita määritellyllä tavalla mutta eivät myöskään ole ristiriidassa niiden sisällön kanssa (Turner & Jain, 2002). Anderson (2005) ehdottaa tähän ratkaisuksi projektissa eteen tulevien ongelmien taltiointia, jolloin myöhemmin kyseisen listan avulla on helpompi määritellä samantyyppisten ongelmien perimmäiset syyt.

5 YHTEENVETO

CMMI:n ja ketterien menetelmien yhteensovittaminen voidaan nähdä kahdelta eri kantilta. Katsantokannasta riippuen se lyhyesti sanottuna joko toimii tai sitten ei. Jos CMMI:n viitekehystä sovelletaan ketteriä menetelmiä käyttävään organisaatioon virallisen kypsyystasoluokituksen saavuttamiseksi, joudutaan joistakin ketteristä periaatteista joustamaan ja lisäämään puuttuvia CMMI:n määrittelemiä toimintoja. Mutta jos tarkoituksena ei ole saada ulkopuolisille tahoille esitettävää todistusaineistoa organisaation kypsyydestä, voidaan CMMI:n määrittelemiä prosessialueita ja ketterien menetelmien parhaita puolia yhdistää huomattavasti vapaammin toimivaksi kokonaisuudeksi, joka jo sinällään tukee organisaation menestymistä markkinoilla.

Saavuttaakseen edes jonkin tason CMMI:n asteikolla ketterin menetelmin, on hyvä ymmärtää, että CMMI tarjoaa neuvoja ja menettelytapoja organisaation laajuisten prosessien hallintaan (Anderson, 2005). Tarkoituksena ei olekaan noudattaa kaikkia ohjeita kirjaimellisesti (Anderson, 2005). Tosin CMMI:n tasolle 2 pääseminen ei tunnu olevan ongelmallista tiukemman linjan ottaneille tulkitsijoillekaan. Mahdolliset ongelma-alueet, joihin tällä tasolla puututtiin, koskivat asioita, joihin kaikki ketterät menetelmät eivät ota kantaa tai eivät täytä täysin vaadittuja prosessialueiden tavoitteita. Yleinen mielipide näyttää kuitenkin tukevan sitä ajatusta, että taso 2 on ketterin menetelmin hyvinkin saavutettavissa. Kähkönen ja Abrahamsson (2004) joutuivat tosin omassa tutkimuksessaan lisäämään muutamia ylimääräisiä toimintoja XP:hen, jotta tason 2 prosessialueet saatiin tyydytettyä täysin. Mutta jos tarkastellaan tarkemmin näitä lisättyjä toimintoja, kuten tiimin päivittäisiä tapaamisia, ”tehtäväkirjaa” ja projektisuunnitelmaa, voidaan huomata niiden yhtäläisyys Scrumissa käytettävien toimintojen kanssa (Kähkönen & Abrahamsson, 2004).

Tason 3, 4 tai 5 saavuttaminen onkin jo vaikeampaa, koska näillä tasoilla eteen tulee myös selviä ristiriitoja joidenkin prosessialueiden ja ketterien menetelmien manifestin välillä. Eniten päänvaivaa aiheuttivat organisaation tasoiset prosessit, jotka vaativat jäykkien toimintamallien noudattamista päätöksenteossa sekä prosessien parantamisessa. Koska tasot 4 ja 5 keskittyvät enemmän organisaation johtamiseen sen prosessien mittaamisen ja analysoimisen keinoin, ketteriä menetelmiä on vaikea niihin soveltaa. Esimerkiksi mittausaineiston saamiseksi *Määrällinen projektin johtaminen* -nimisessä prosessialueessa on kerättävä paljon tietoa mitattavien prosessien päämääristä ja erityisesti näiden päämäärien saavuttamisesta (Software Engineering Institute, 2006b). Näiden tietojen pohjalta organisaatio pystyy määrittelemään prosessinsa tarkasti ja valvomaan niiden toteutumista (Software Engineering Institute, 2006b). Mutta tarkkaan edeltä määriteltyjen prosessien noudattaminen ei ole samalla linjalla ketterien menetelmien manifestin kanssa, joka peräänkuuluttaa prosessien noudattamisen sijaan yksilöiden ja vuorovaikutuksen tärkeyttä (Beck ym., 2001). Muutamassa artikkelissa näitäkään ongelmia ei ole pidetty ylitsepääsemättöminä, mm. Anderson (2005) sekä Sutherland, Ruseng Jakobsen ja Johnson (2008) näkevät tason 3 olevan hyvinkin ketterin menetelmin saavutettavissa, eikä tasoille 4 tai 5 pääseminenkään ole täysin mahdoton ajatus.

Huomattavia eroja tutkielmassa käytettyjen artikkelien välillä ei juuri ollut. Eniten mielipide-eroja tuntui syntyvän riippuen käytettävästä ketterästä menetelmästä. XP:tä voidaan luonnehtia menetelmänä enemmän ohjelmointi-painotteiseksi, kun taas Scrumissa otetaan huomioon paremmin myös ohjelmistoprojektien hallinnalliset asiat (Marcal ym., 2008). Tästä johtuen Scrum näyttäisi olevan kokonaisuutena soveltuvampi menetelmä yhdistettäväksi CMMI:n kanssa. Tälle yhdistämiselle on tulevaisuudessakin varmasti tarvetta, koska monissa empiirisissä tutkimuksissa todettiin Sutherlandin ym. (2008) tavoin niiden tukevan toisiaan. CMMI tarjoaa organisaatiolle kurinalaisen näkökulman sen prosessien

parantamiseen ja pitää huolen, että kaikki tarpeellinen on otettu huomioon, kun taas ketterät menetelmät tuovat tähän kurinalaisuuteen ripauksen mukautumiskykyä ja liikearvoa (Sutherland, Ruseng Jakobsen, & Johnson, 2008).

Myös CMMI:n kehittäjäorganisaatio Software Engineering Institute (SEI) on ottanut kantaa ketterien menetelmien ja CMMI:n yhdistämiseen. Syiksi yhdistämisen epäonnistumiselle he luettelevat mm. käytettyjen termien väärinymmärryksen, oikean tiedon puuttumisen ja CMMI:n mallin väärinkäytön (Software Engineering Institute, 2008a). Juuri tällaisiin yleisiin väärinymmärryksiin kannattaisikin kiinnittää huomiota vaikkapa kehittämällä selkeämpi, myös ketterät menetelmät paremmin huomioon ottava ohjelmistokehityksen kypsyystasomalli. Tärkeintä tässä uuden mallin kehitysprosessissa olisi ottaa mukaan myös ketterien menetelmien ammattilaisia, jotta voitaisiin saavuttaa sellainen viitekehys, jota kaikki osapuolet osaisivat tulkita samalla tavalla. Tällainen kehitysprosessi ei tapahdu aivan hetkessä, koska sekä CMMI:n että ketterien menetelmien tulkintoja tuntuu olevan yhtä monta kuin tulkitsijoitakin.

Tutkielman heikkoutena voidaan nähdä riittävän tietotaidon puute. Koska vertailussa käytettävät artikkelit käsittelevät eri ketteriä menetelmiä sekä CMM:n tai CMMI:n eri versioita, kunkin artikkelin kriittinen tarkastelu on huomattavan vaikeaa. Tutkielman tavoitteena on kuitenkin ollut löytää näistä artikkeleista sellaisia argumentteja ja tutkimustuloksia joiden voidaan ajatella olevan verrattavissa keskenään. Tämän tutkielman puitteissa aukoton ja yleistettävissä oleva vertailu ei ole kuitenkaan käytännössä mahdollista, vaan se vaatisi syvempää asiantuntijuutta sekä CMMI:n että ketterien menetelmien osalta. Aiheessa riittää sijaa myös jatkotutkimukselle, ennen kuin saavutetaan yleisesti hyväksytty tasapaino ketterien menetelmien ja CMMI:n välille. Ehkä ajan myötä, kun saadaan lisää empiiristä tutkimusta myös yhdistämisen ongelmallisista alueista, saadaan hiottua esiin toisiaan tukevat prosessit ja menetelmät

molemmista osapuolista. Yhteisen sävelen löytäminen CMMI:n ja ketterien menetelmien välille toisi enemmän hyötyä organisaatiolle kuin kummankaan käyttö toisi yksinään (Sutherland ym., 2008). Tässä tapauksessa yksi plus yksi onkin enemmän kuin kaksi.

LÄHTEET

Abrahamsson, P., Salo, O., & Ronkainen, J. (2002). *Agile software development methods: Review and analysis*. Espoo [Finland]: VTT.

Anderson, D. J. (2005). *Stretching agile to fit CMMI level 3 - the story of creating MSF for CMMI; process improvement at microsoft corporation.* Agile Conference, 2005. Proceedings, 193-201.

Beck, K., ym. (2001). *Agile manifesto.* Retrieved October 24, 2008, from http://agilemanifesto.org/

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

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

Fritzsche, M., & Keil, P. (2007). *Agile methods and CMMI: Compatibility or conflict?* E-Informatica Software Engineering Magazine, 1(1), 9-26.

Kähkönen, T., & Abrahamsson, P. (2004). *Achieving CMMI level 2 with enhanced extreme programming approach*. Product Focused Software Process Improvement, 2004. Proceedings, 3009 378-392.

Marcal, A., De Freitas, B., Furtado Soares, F., Marciel, T., & Belchior, A. (2008). *Blending scrum practices and CMMI project management process areas*. Innovations in Systems and Software Engineering, 4(1), 17-29.

Marcal, A., de Freitas, B., Furtado Soares, F., & Belchior, A. (2007). *Mapping CMMI project management process areas to SCRUM practices.* Software Engineering Workshop, 2007. SEW 2007. 31st IEEE, 2007, 13-22.

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

Paulk, M., Curtis, B., Chrissis, M., & Weber, C. (1993). *Capability maturity model, version 1.1.* IEEE Software. Vol.10, 10(4), 18-27.

Pikkarainen, M. (2008). *Towards a framework for improving software development process mediated with CMMI goals and agile practices* No. VTT Publications 695. Espoo [Finland]: VTT.

Pikkarainen, M., & Mäntyniemi, A. (2006). *An approach for using CMMI in agile software development assessments: Experiences from three case studies.* SPICE Conference, 2006.

Schwaber, K. (1995). *SCRUM development process*. Paper presented at the OOPSLA'95 Workshop on Business Object Design and Implementation, 2008(11/14) 23. Retrieved from http://jeffsutherland.com/oopsla/schwapub.pdf

Software Engineering Institute. (2001). *Capability maturity model integration (CMMI), version 1.1.* Carnegie Mellon University. Retrieved from http://www.fastwrite.com/dharma/02tr003.pdf

Software Engineering Institute. (2006a). *Appraisal requirements for CMMI, version 1.2 (ARC, V1.2).* Carnegie Mellon University. Retrieved from http://www.sei.cmu.edu/pub/documents/06.reports/pdf/06tr011.pdf

Software Engineering Institute. (2006b). *CMMI for development, version 1.2.* Carnegie Mellon University. Retrieved from http://www.sei.cmu.edu/pub/documents/06.reports/pdf/06tr008.pdf

Software Engineering Institute. (2007a). *Capability maturity model integration (CMMI) version 1.2 overview.* Retrieved 11/25, 2008, from http://www.sei.cmu.edu/cmmi/adoption/pdf/cmmi-overview07.pdf

Software Engineering Institute. (2007b). *CMMI for acquisition, version 1.2.* Carnegie Mellon University. Retrieved from http://www.sei.cmu.edu/pub/documents/07.reports/07tr017.pdf

Software Engineering Institute. (2008a). *CMMI and agile: Why not embrace both!* Carnegie Mellon University. Retrieved from http://www.sei.cmu.edu/pub/documents/08.reports/08tn003.pdf

Software Engineering Institute. (2008b). *CMMI for services (CMMI-SVC), partner and piloting draft, V0.9c.* Carnegie Mellon University. Retrieved from http://www.sei.cmu.edu/cmmi/models/CMMI-Services-status.html

Sutherland, J. (2004). *Agile development: Lessons learned from the first scrum.* Retrieved 11/18, 2008, from http://jeffsutherland.com/scrum/FirstScrum2004.pdf

Sutherland, J., Ruseng Jakobsen, C., & Johnson, K. (2008). *Scrum and CMMI level 5: The magic potion for code warriors.* Hawaii International Conference on System Sciences, Proceedings of the 41st Annual, 466-466.

Turner, R., & Jain, A. (2002). *Agile meets CMMI: Culture clash or common cause?* Kirjasta Extreme programming and agile methods - XP/Agile universe 2002, (pp. 153-165)

Vanderburg, G. (2005). *A simple model of agile software processes - or - extreme programming annealed.* ACM SIGPLAN NOTICES, 40(10), 539-545.

LIITE 1: CMMI-DEV PROSESSIALUEET

Suomennettu lähteestä Software Engineering Institute, Carnegie Mellon University; 2006, CMMI for Development, Version 1.2

TASO 1

Tällä tasolla ei ole erityisiä tavoitteita tai toimintatapoja.

TASO 2

Toimittajasopimusten hallinta

Supplier Agreement Management (SAM)
Osoittaa projektien tarpeen omaksua ne osat työstä, jotka toimittajat ovat tehneet. Toimittajan edistymistä ja suorituskykyä seurataan valvomalla työn tuotoksia ja työprosesseja, toimittajasopimusta tarkastellaan uudelleen tarvittaessa. Toimittajan tekemät komponentit arvioidaan ja testataan.

SG 1 Toimittajasopimusten laatiminen (Establish Supplier Agreements)
- SP 1.1 Määritä hankinnan tyyppi (Determine Acquisition Type)
- SP 1.2 Valitse toimittajat (Select Suppliers)
- SP 1.3 Laadi toimittajasopimukset (Establish Supplier Agreements)

SG 2 Toimittajasopimusten täyttäminen (Satisfy Supplier Agreements)
- SP 2.1 Toimittajasopimuksen toimeenpano (Execute the Supplier Agreement)
- SP 2.2 Tarkkaile valittuja toimittajaprosesseja (Monitor Selected Supplier Processes)
- SP 2.3 Arvioi valittuja toimittajien työn tuotteita (Evaluate Selected Supplier Work Products)
- SP 2.4 Hyväksy toimittajasopimuksen mukaiset hankitut tuotteet (Accept the Acquired Product)
- SP 2.5 Tuotteiden siirtäminen toimittajalta (Transition Products)

Vaatimusten hallinta

Requirements Management (REQM)
Kuvailee vaatimusten kontrollointiin ja hankkimiseen sekä asiaankuuluvien suunnitelmien ja aineiston ajantasalla pitämiseen liittyviä toimintoja. Tämän prosessialueen ansiosta voidaan jäljittää asiakkaan vaatimukset tuotteen komponenttien tasolle asti.

SG 1 Hallitse vaatimuksia (Manage Requirements)
- SP 1.1 Pyri ymmärtämään vaatimuksia (Obtain an Understanding of Requirements)
- SP 1.2 Sitoudu vaatimusten noudattamiseen (Obtain Commitment to Requirements)
- SP 1.3 Hallitse vaatimusten muutokset (Manage Requirements Changes)
- SP 1.4 Säilytä vaatimusten jäljitettävyys molempiin suuntiin (Maintain Bidirectional Traceability of Requirements)
- SP 1.5 Tunnista projektin työn ja vaatimusten väliset epäjohdonmukaisuudet (Identify Inconsistencies Between Project Work and Requirements)

Projektisuunnittelu

Project Planning (PP)
Prosessialue käsittelee projektisuunnitelman kehittämistä, osakkaiden tarkoituksenmukaista mukanaoloa, suunnitelmaan sitoutumista ja suunnitelmassa pitäytymistä.

SG 1 Arvioinnin laatiminen (Establish Estimates)
- SP 1.1 Arvioi projektin laajuus (Estimate the Scope of the Project)
- SP 1.2 Laadi työn tuotoksen ja tehtävien ominaisuuksien arviointi (Establish Estimates of Work Product and Task Attributes)
- SP 1.3 Määrittele projektin elinkaari (Define Project Lifecycle)
- SP 1.4 Päätä työn ja kustannusten arviointikriteereistä (Determine Estimates of Effort and Cost)

SG 2 Kehitä projektisuunnitelma (Develop a Project Plan)
- SP 2.1 Laadi budjetti ja aikataulu (Establish the Budget and Schedule)
- SP 2.2 Tunnista projektin riskit (Identify Project Risks)
- SP 2.3 Suunnitelma datan hallintaa varten (Plan for Data Management)
- SP 2.4 Suunnitelma projektin resursseja varten (Plan for Project Resources)
- SP 2.5 Määrittele tarvittavat tietotaitot ja kyvyt (Plan for Needed Knowledge and Skills)
- SP 2.6 Määrittele asianosaisten osallistuminen (Plan Stakeholder Involvement)
- SP 2.7 Laadi projektisuunnitelma (Establish the Project Plan)

SG 3 Suunnitelmaan sitoutuminen (Obtain Commitment to the Plan)
- SP 3.1 Tarkastele projektiin vaikuttavat suunnitelmat läpi (Review Plans That Affect the Project)
- SP 3.2 Sovita työn ja resurssin tasot toisiinsa (Reconcile Work and Resource Levels)
- SP 3.3 Pidä huolta, että suunnitelmaan sitoudutaan. (Obtain Plan Commitment)

Projektin valvonta ja hallinta

Project Monitoring and Control (PMC)
Toimintojen valvontaa ja tarvittaessa korjaavien toimien käyttöönottoa. Projektisuunnitelmassa päätetään, kuinka tarkasti projektia valvotaan, kuinka usein edistymistä tarkistetaan, sekä kuinka edistymisen valvonta toteutetaan.

SG 1 Tarkkaile projektia suunnitelman näkökulmasta (Monitor Project Against Plan)

- SP 1.1 Tarkkaile projektin suunnittelun parametreja (Monitor Project Planning Parameters)
- SP 1.2 Tarkkaile sitoutumista (Monitor Commitments)
- SP 1.3 Tarkkaile projektin riskejä (Monitor Project Risks)
- SP 1.4 Tarkkaile datan hallintaa (Monitor Data Management)
- SP 1.5 Tarkkaile asianosaisten osallistumista (Monitor Stakeholder Involvement)
- SP 1.6 Johda edistymisen tarkasteluja (Conduct Progress Reviews)
- SP 1.7 Johda virstanpylväiden tarkasteluja (Conduct Milestone Reviews)

SG 2 Huolehdi korjaustoimet loppuun saakka (Manage Corrective Action to Closure)

- SP 2.1 Analysoi ongelmat (Analyze Issues)
- SP 2.2 Tee korjaustoimet (Take Corrective Action)
- SP 2.3 Ohjaa korjaustoimia (Manage Corrective Action)

Kokoonpanon hallinta

Configuration Management (CM)
Tukee kaikkia prosessialueita vakiinnuttamalla ja ylläpitämällä ehyttä työn tuotetta hyödyntämällä kokoonpanon tunnistamista, kontrollointia, tilanteen kirjanpitoa ja seurantaa. Esimerkkejä työn tuotteista, jotka voidaan käsitellä tämän prosessialueen avulla sisältävät mm. suunnitelmat, menetelmänkuvaukset, vaatimukset, suunnitelmadatan, piirustukset, tuotteen erittelyt, koodin, kääntäjät, tuotteen datatiedostot, sekä tuotteen tekniset julkaisut.

SG 1 Luo lähtökohdat (Establish Baselines)

- SP 1.1 Tunnista kokoonpanon yksityiskohdat (Identify Configuration Items)
- SP 1.2 Luo kokoonpanon hallintajärjestelmä (Establish a Configuration Management System)
- SP 1.3 Luo tai vapauta lähtökohdat (Create or Release Baselines)

SG 2 Jäljitä ja kontrolloi muutokset (Track and Control Changes)

- SP 2.1 Jäljitä muutospyynnöt (Track Change Requests)
- SP 2.2 Kontrolloi kokoonpanon yksityiskohtia (Control Configuration Items)

SG 3 Luo eheä kokonaisuus (Establish Integrity)

- SP 3.1 Luo kokoonpanon hallinta-arkisto (Establish Configuration Management Records)
- SP 3.2 Suorita kokoonpanon tarkastuksia (Perform Configuration Audits)

Mittaus ja analyysi

Measurement and Analysis (MA)
Mittaus ja analyysi tukee kaikkia muita prosessialueita tarjoamalla tiettyjä toimintatapoja projektin ja organisaation opastamiseen kun sovitetaan

mittaustarvetta ja –päämäärää puolueettomia tuloksia tarjoavaan mittaustapaan.

SG 1 Linjaa mittaus- ja analyysitoiminnot tukemaan toisiaan (Align Measurement and Analysis Activities)

- SP 1.1 Luo mittaustavoitteet (Establish Measurement Objectives)
- SP 1.2 Täsmennä toimenpiteet (Specify Measures)
- SP 1.3 Täsmennä datan keruun ja varastoinnin menettelytavat (Specify Data Collection and Storage Procedures)
- SP 1.4 Täsmennä analyysin menettelytavat (Specify Analysis Procedures)

SG 2 Järjestä mittaustuloksia (Provide Measurement Results)

- SP 2.1 Kerää mittausdataa (Collect Measurement Data)
- SP 2.2 Analysoi mittausdataa (Analyze Measurement Data)
- SP 2.3 Arkistoi data ja tulokset (Store Data and Results)
- SP 2.4 Välitä tulokset eteenpäin (Communicate Results)

Prosessin ja tuotteen laadunvarmistus

Process and Product Quality Assurance (PPQA)
Tarjoaa toimintatapoja suoritettujen prosessien, sekä työn tuotteiden ja palvelujen objektiiviseen arviointiin kun niitä verrataan soveltuviin prosessin määrittelyihin, standardeihin ja menettelytapoihin. Jos näissä katselmuksissa nousee esille jotakin ongelmia, tämä prosessialue pitää huolen siitä, että niihin puututaan.

SG 1 Arvioi prosesseja ja työn tuotoksia tasapuolisesti (Objectively Evaluate Processes and Work Products)

- SP 1.1 Arvioi prosesseja tasapuolisesti (Objectively Evaluate Processes)
- SP 1.2 Arvioi työn tuotoksia ja palveluja tasapuolisesti (Objectively Evaluate Work Products and Services)

SG 2 Tarjoa tasapuolisesti tietoa (Provide Objective Insight)

- SP 2.1 Välitä ja varmista ratkaisut poikkeaviin tilanteisiin (Communicate and Ensure Resolution of Noncompliance Issues)
- SP 2.2 Perusta arkisto (Establish Records)

TASO 3

Organisaation prosessien määrittely

Organizational Process Definition +IPPD (OPD+IPPD)
Prosessialueen tarkoituksena perustaa ja säilyttää organisaation standardiprosessit, työympäristön standardit ja muut organisaation edut ja päämäärät.

SG 1 Luo organisaation prosessin apukeinot (Establish Organizational Process Assets)

- SP 1.1 Luo standardiprosessit (Establish Standard Processes)
- SP 1.2 Luo elinkaarimallin kuvaus (Establish Lifecycle Model Descriptions)
- SP 1.3 Luo räätälöinnin kriteerit ja suositukset (Establish Tailoring Criteria and Guidelines)
- SP 1.4 Luo organisaation mittausrepositorio (Establish the Organization's Measurement Repository)
- SP 1.5 Luo organisaation prosessien apukeinojen kirjasto (Establish the Organization's Process Asset Library)
- SP 1.6 Luo työympäristön standardit (Establish Work Environment Standards)

Prosessit organisaation näkökulmasta

Organizational Process Focus (OPF)
Auttaa organisaatiota suunnittelemaan, toteuttamaan ja ottamaan käyttöön organisaation laajuisia prosessien parannuksia, jotka pohjautuvat olemassa olevien prosessien ja prosessien apukeinojen vahvuuksien ja heikkouksien ymmärtämiseen.

SG 1 Määrittele prosessin parantamisen mahdollisuudet (Determine Process Improvement Opportunities)

- SP 1.1 Luo organisaation prosessien tarpeet (Establish Organizational Process Needs)
- SP 1.2 Arvioi organisaation prosessit (Appraise the Organization's Processes)
- SP 1.3 Identifioi organisaation prosessien parannukset (Identify the Organization's Process Improvements)

SG 2 Suunnittele ja toteuta prosessien parannukset (Plan and Implement Process Improvements)

- SP 2.1 Luo prosessien toimenpiteiden suunnitelmat (Establish Process Action Plans)
- SP 2.2 Toteuta prosessien toimenpiteiden suunnitelmat (Implement Process Action Plans)

SG 3 Käytä hyväksi organisaation prosessien apukeinot ja liitä mukaan opitut asiat (Deploy Organizational Process Assets and Incorporate Lessons Learned)

- SP 3.1 Käytä hyväksi organisaation prosessien apukeinot (Deploy Organizational Process Assets)
- SP 3.2 Käytä hyväksi standardoidut prosessit (Deploy Standard Processes)
- SP 3.3 Tarkkaile toteutusta (Monitor Implementation)
- SP 3.4 Ota prosesseihin liittyvät kokemukset mukaan organisaation apukeinoihin (Incorporate Process-Related Experiences into the Organizational Process Assets)

Organisaation kouluttaminen

Organizational Training (OT)
Identifioi organisaation strategiset koulutuksen tarpeet sekä taktiset, yleiset koulutuksen tarpeet, joita ilmenee organisaation projekteissa ja tukiryhmissä.

SG 1 Perusta organisaation koulutusvoimavarat (Establish an Organizational Training Capability)

- SP 1.1 Luo strategiset koulutuksen tarpeet (Establish the Strategic Training Needs)
- SP 1.2 Määrittele organisaation vastuulla olevat koulutustarpeet (Determine Which Training Needs Are the Responsibility of the Organization)
- SP 1.3 Luo organisaation koulutuksen taktinen suunnitelma (Establish an Organizational Training Tactical Plan)
- SP 1.4 Perusta koulutusvoimavarat (Establish Training Capability)

SG 2 Tarjoa tarpeellinen koulutus (Provide Necessary Training)

- SP 2.1 Toimita koulutus (Deliver Training)
- SP 2.2 Luo koulutusarkisto (Establish Training Records)
- SP 2.3 Määritä koulutuksen tehokkuus (Assess Training Effectiveness)

Integroitu projektin johtaminen

Integrated Project Management +IPPD (IPM+IPPD)
Projektille määritellyn prosessin perustaminen ja ylläpito. Prosessit räätälöidään kunkin projektin erikoistarpeet huomioon ottaen käyttäen lähtökohtana organisaation standardiprosesseja.
Kun integroidun projektin johtamisen perään lisätään vielä IPPD (lyhenne sanoista Integrated Product and Process Development), kyse on integroidusta tuotteen ja prosessien kehityksestä, tehtävästä joka koskee useampaa tiimiä. Tällöin yhteisen näkemyksen luominen ja tiimien riittävä yhteistyö on tärkeää.

SG 1 Käytä projektin määriteltyjä prosesseja (Use the Project's Defined Process)

- SP 1.1 Perusta projektin määritelty prosessi (Establish the Project's Defined Process)
- SP 1.2 Käytä hyväksi organisaation apukeinoja suunnitellessasi projektien toimintoja (Use Organizational Process Assets for Planning Project Activities)
- SP 1.3 Perusta projektin työympäristö (Establish the Project's Work Environment)
- SP 1.4 Yhdistä suunnitelmat (Integrate Plans)
- SP 1.5 Hallinnoi projektia yhdistettyjen suunnitelmien avulla (Manage the Project Using the Integrated Plans)
- SP 1.6 Edistä organisaation prosessien apukeinoja (Contribute to the Organizational Process Assets)

SG 2 Koordinoi ja tee yhteistyötä osakkaiden kanssa (Coordinate and Collaborate with Relevant Stakeholders)

- SP 2.1 Johda osakkaiden osallistumista (Manage Stakeholder Involvement)
- SP 2.2 Hallinnoi riippuvuussuhteita (Manage Dependencies)
- SP 2.3 Ratkaise koordinointiongelmat (Resolve Coordination Issues)

Riskienhallinta

Risk Management (RSKM)

Riskienhallinta on eteenpäinsuuntautunutta ja jatkuvaa. Sen aktiviteetit sisältävät riskimuuttujien tunnistamisen, riskien arvioinnin ja niiden lieventämisen.

SG 1 Valmistaudu riskienhallintaan Prepare for Risk Management)
- SP 1.1 Määrittele riskien lähteet ja kategoriat (Determine Risk Sources and Categories)
- SP 1.2 Määrittele riskien muuttujat (Define Risk Parameters)
- SP 1.3 Luo riskienhallinnan strategia (Establish a Risk Management Strategy)

SG 2 Tunnista ja analysoi riskit (Identify and Analyze Risks)
- SP 2.1 Tunnista riskit (Identify Risks)
- SP 2.2 Arvioi, luokittele ja laita riskit tärkeysjärjestykseen (Evaluate, Categorize, and Prioritize Risks)

SG 3 Vähennä riskejä (Mitigate Risks)
- SP 3.1 Kehitä riskien vähentämissuunnitelmia (Develop Risk Mitigation Plans)
- SP 3.2 Toteuta riskien vähentämissuunnitelmat (Implement Risk Mitigation Plans)

Vaatimusten kehittäminen

Requirements Development (RD)
Identifioi asiakkaiden tarpeet ja muokkaa ne tuotteen vaatimuksiksi. Tällä tavoin syntyneet tuotteen ja sen komponenttien vaatimukset kuvailevat selkeästi ohjelmiston kehittäjän ymmärtämillä ja käyttämillä termeillä tuotteen suorituskykyä, suunnittelun erityispiirteitä, varmennusvaatimuksia jne.

SG 1 Kehitä asiakkaan vaatimukset (Develop Customer Requirements)
- SP 1.1 Paljasta tarpeet (Elicit Needs)
- SP 1.2 Kehitä asiakkaan vaatimukset (Develop the Customer Requirements)

SG 2 Kehitä tuotteen vaatimukset (Develop Product Requirements)
- SP 2.1 Perusta tuotteen ja tuotteen komponenttien vaatimukset (Establish Product and Product Component Requirements)
- SP 2.2 Kohdenna tuotteen komponenttien vaatimukset (Allocate Product Component Requirements)
- SP 2.3 Tunnista käyttöliittymän vaatimukset (Identify Interface Requirements)

SG 3 Analysoi ja vahvista vaatimukset (Analyze and Validate Requirements)
- SP 3.1 Perusta toiminnalliset käsitteet ja skenaariot (Establish Operational Concepts and Scenarios)
- SP 3.2 Luo määrittelyt vaadituille toiminnallisuuksille (Establish a Definition of Required Functionality)
- SP 3.3 Analysoi vaatimukset (Analyze Requirements)
- SP 3.4 Analysoi vaatimukset saavuttaaksesi tasapainon (Analyze Requirements to Achieve Balance)
- SP 3.5 Vahvista vaatimukset (Validate Requirements)

Tekninen ratkaiseminen

Technical Solution (TS)
Tämä prosessialue kehittää teknisiä datapaketteja tuotteen komponenteille, joita PI tai SAM -prosessialueet käyttävät.

SG 1 Valitse tuotteen komponenttien ratkaisut (Select Product Component Solutions)
- SP 1.1 Kehitä vaihtoehtoisia ratkaisuja ja valintakriteerejä (Develop Alternative Solutions and Selection Criteria)
- SP 1.2 Valitse tuotteen komponenttien ratkaisut (Select Product Component Solutions)

SG 2 Kehitä tuotesuunnittelua (Develop the Design)
- SP 2.1 Suunnittele tuote tai tuotteen komponentti (Design the Product or Product Component)
- SP 2.2 Perusta tekninen datapaketti (Establish a Technical Data Package)
- SP 2.3 Suunnittele käyttöliittymät käyttäen kriteereitä (Design Interfaces Using Criteria)
- SP 2.4 Suorita, tee, osta tai uudelleenkäytä analyysejä (Perform Make, Buy, or Reuse Analyses)

SG 3 Toteuta tuotteen suunnittelu (Implement the Product Design)
- SP 3.1 Toteuta tuotesuunnittelu (Implement the Design)
- SP 3.2 Kehitä tuotteen tukidokumentaatio (Develop Product Support Documentation)

Tuotteen integrointi

Product Integration (PI)
Tuotteen integroinnin prosessialue sisältää parhaaseen mahdolliseen integraation jaksoon, tuotteen komponenttien integrointiin, sekä tuotteen asiakkaalle toimitukseen liittyviä käytäntöjä.

SG 1 Valmistaudu tuotteen integrointiin (Prepare for Product Integration)
- SP 1.1 Määrittele integraation jaksot (Determine Integration Sequence)
- SP 1.2 Luo tuotteen integraatioympäristö (Establish the Product Integration Environment)
- SP 1.3 Luo tuotteen integraation menettelytavat ja kriteerit (Establish Product Integration Procedures and Criteria)

SG 2 Varmista käyttöliittymän yhteensopivuus (Ensure Interface Compatibility)
- SP 2.1 Tarkista käyttöliittymän kuvaukset löytyvyyden takaamiseksi (Review Interface Descriptions for Completeness)
- SP 2.2 Hallitse käyttöliittymiä (Manage Interfaces)

SG 3 Kokoa tuotteen komponentit ja toimita tuote (Assemble Product Components and Deliver the Product)
- SP 3.1 Vahvista tuotteiden komponenttien valmiuden integraatiota varten (Confirm Readiness of Product Components for Integration)
- SP 3.2 Kokoa tuotteen komponentit (Assemble Product Components)
- SP 3.3 Arvioi kootut tuotteen komponentit (Evaluate Assembled Product Components)

SP 3.4 Paketoi ja toimita tuote tai tuotteen komponentti (Package and Deliver the Product or Product Component)

Validointi

Validation (VAL)
Tämä prosessialue validoi tuotteita peilaten niitä asiakkaan tarpeisiin. Se, että validointivaatimukset kehitetään yhteistyössä asiakkaan kanssa on tärkeä elementti tässä prosessialueessa.
Validoinnin kohteita ovat mm. tuotteet, niiden komponentit, sekä valikoidut välivaiheen tuotteet ja prosessit.

SG 1 Valmistaudu validointia varten (Prepare for Validation)
- SP 1.1 Valitse tuotteet validointiin (Select Products for Validation)
- SP 1.2 Luo validointiympäristö (Establish the Validation Environment)
- SP 1.3 Luo validointimenettelytavat ja –kriteerit (Establish Validation Procedures and Criteria)

SG 2 Validoi tuote tai tuotteen komponentit (Validate Product or Product Components)
- SP 2.1 Toteuta validointi (Perform Validation)
- SP 2.2 Analysoi validoinnin tulokset (Analyze Validation Results)

Varmennus

Verification (VER)
Prosessialue pitää huolen siitä, että valitut työn tuotteet ovat niille määriteltyjen vaatimusten mukaisia. Varmennus on yleensä vähittäinen prosessi, alkaen tuotteen komponenttien varmentamisella ja yleensä loppuen kokonaan kasatun tuotteen varmentamiseen.
Varmennuksen yhtenä osana on myös vertaisarviointi. Vertaisarvioinnit on todistetusti tehokas menetelmä virheiden huomaamiseen sekä poistamiseen varhaisessa vaiheessa ja ne mahdollistavat arvokkaan näkökulman työn tuotteisiin ja niiden komponentteihin.

SG 1 Valmistaudu varmennusta varten (Prepare for Verification)
- SP 1.1 Valitse työn tuotteet varmennusta varten (Select Work Products for Verification)
- SP 1.2 Luo varmennusympäristö (Establish the Verification Environment)
- SP 1.3 Luo varmennusmenettelytavat ja –kriteerit (Establish Verification Procedures and Criteria)

SG 2 Suorita vertaisarviointeja (Perform Peer Reviews)
- SP 2.1 Valmistaudu vertaisarviointia varten (Prepare for Peer Reviews)
- SP 2.2 Suorita vertaisarvioinnit (Conduct Peer Reviews)
- SP 2.3 Analysoi vertaisarvioinnin data (Analyze Peer Review Data)

SG 3 Varmenna valitut työn tuotokset (Verify Selected Work Products)
- SP 3.1 Suorita varmennus (Perform Verification)

SP 3.2 Analysoi varmennuksen tulokset (Analyze Verification Results)

Päätöksenteon analyysi ja ratkaiseminen

Decision Analysis and Resolution (DAR)
Tämä prosessialue tukee kaikkia prosessialueita määrittelemällä, mitkä ongelmat pitäisi käydä läpi virallisen arviointiprosessin avulla.

SG 1 Arvioi vaihtoehdot (Evaluate Alternatives)

SP 1.1 Luo päätöksenteon analyysin suuntaviivat (Establish Guidelines for Decision Analysis)
SP 1.2 Luo arviointikriteerit (Establish Evaluation Criteria)
SP 1.3 Identifioi vaihtoehtoiset ratkaisut (Identify Alternative Solutions)
SP 1.4 Valitse arviointimenetelmät (Select Evaluation Methods)
SP 1.5 Arvioi vaihtoehdot (Evaluate Alternatives)
SP 1.6 Valitse ratkaisut (Select Solutions)

TASO 4

Organisaation prosessin suorituskyky

Organizational Process Performance (OPP)
Organisaation liiketoimintatavoitteista saadaan määrällisiä tavoitteita laadun ja prosessin suorittamiseen. Organisaatio tarjoaa projekteille ja tukiryhmille yleisiä mittapuita, prosessin suorituskyvyn lähtökohtia sekä prosessin suorituskykymalleja.

SG 1 Luo suorituskyvyn lähtökohdat ja mallit (Establish Performance Baselines and Models)

SP 1.1 Valitse prosessit (Select Processes)
SP 1.2 Luo prosessin suorituskyvyn mittakaava (Establish Process-Performance Measures)
SP 1.3 Laadi laadun ja prosessien suorituskyvyn tavoitteet (Establish Quality and Process-Performance Objectives)
SP 1.4 Laadi prosessin suorituskyvyn lähtökohdat (Establish Process-Performance Baselines)
SP 1.5 Laadi prosessin suorituskyvyn mallit (Establish Process-Performance Models)

Määrällinen projektin johtaminen

Quantitative Project Management (QPM)
Soveltaa määrällisiä ja tilastollisia tekniikoita prosessin suorituskyvyn hallintaan ja tuotteiden laadunvarmistukseen.

SG 1 Johda projektia määrällisesti (Quantitatively Manage the Project)

SP 1.1 Laadi projektin tavoitteet (Establish the Project's Objectives)
SP 1.2 Laadi määritelty prosessi (Compose the Defined Process)

SP 1.3 Valitse aliprosessit, joita hallinnoidaan tilastollisin keinoin (Select the Subprocesses that Will Be Statistically Managed)
SP 1.4 Hallinnoi projektin suorituskykyä (Manage Project Performance)

SG 2 Hallinnoi tilastollisin keinoin aliprosessien suorituskykyä (Statistically Manage Subprocess Performance)
SP 2.1 Valitse mittajärjestelmä ja analyyttiset tekniikat (Select Measures and Analytic Techniques)
SP 2.2 Käytä tilastollisia menetelmiä ymmärtääksesi vaihtelua (Apply Statistical Methods to Understand Variation)
SP 2.3 Tarkkaile valittujen aliprosessien suorituskykyä (Monitor Performance of the Selected Subprocesses)
SP 2.4 Tallenna tilastollinen hallinnon data (Record Statistical Management Data)

TASO 5

Organisaation innovaatio ja järjestäytyminen

Organizational Innovation and Deployment (OID)
Tämä prosessialue valikoi ja käyttää hyödykseen ehdotettuja vähittäisiä ja innovatiivisia parannuksia, jotka vaikuttavat organisaation kykyyn vastata lupaamiinsa laadullisiin ja prosessin suorituskyvyllisiin tavoitteisiin. Käyttöönotettavien parannusten valitseminen perustuu kvantitatiiviseen ymmärrykseen todennäköisistä hyödyistä ja kustannuksista.

SG 1 Valitse parannukset (Select Improvements)
SP 1.1 Kerää ja analysoi kehitysehdotukset (Collect and Analyze Improvement Proposals)
SP 1.2 Tunnista ja analysoi innovaatiot (Identify and Analyze Innovations)
SP 1.3 Testaa parannukset (Pilot Improvements)
SP 1.4 Valitse käyttöön otettavat parannukset (Select Improvements for Deployment)

SG 2 Ota käyttöön parannukset (Deploy Improvements)
SP 2.1 Suunnittele käyttöönotto (Plan the Deployment)
SP 2.2 Hallinnoi käyttöönottoa (Manage the Deployment)
SP 2.3 Mittaa parannusten vaikutukset (Measure Improvement Effects)

Syysuhteen analyysi ja ratkaiseminen

Causal Analysis and Resolution (CAR)
Tämän prosessialueen avulla projektin jäsenet tunnistavat valikoitujen vikojen syyt ja muut ongelmat sekä toimivat niiden ehkäisemiseksi tulevaisuudessa.

SG 1 Määrittele virheiden aiheuttajat (Determine causes of defects)
SP 1.1 Valitse dataa vioista analyysiä varten (Select defect data for analysis)
SP 1.2 Analysoi aiheuttajat (Analyze causes)

SG 2 Keskity virheiden aiheuttajiin (Address causes of defects)

SP 2.1 Ota käyttöön toimintaehdotukset (Implement the action proposals)

SP 2.2 Arvioi muutosten vaikutukset (Evaluate the effect of changes)

SP 2.3 Tallenna tiedot (Record data)

LIITE 2: CMMI-DEV YLEISET TAVOITTEET

Suomennettu lähteestä Software Engineering Institute, Carnegie Mellon University; 2006, CMMI for Development, Version 1.2

GG 1 Saavuta erityistavoitteet (Achieve Specific Goals)

Prosessi tukee ja mahdollistaa prosessialueen erityistavoitteiden saavuttamisen muuntamalla tunnistettavat syötettävät työn tuotokset tunnistettaviksi toimituksen työn tuotoksiksi.

GP 1.1 Suorita erityistoimet (Perform Specific Practices)

GG 2 Vakiinnuta hallittu prosessi (Institutionalize a Managed Process)

Prosessi on vakiintunut hallittuna prosessina.

GP 2.1 Luo organisaation laajuinen menettelytapa (Establish an Organizational Policy)
GP 2.2 Suunnittele prosessin kulku (Plan the Process)
GP 2.3 Järjestä resurssit (Provide Resources)
GP 2.4 Aseta vastuualueet (Assign Responsibility)
GP 2.5 Kouluta ihmiset (Train People)
GP 2.6 Hallitse kokoonpanot (Manage Configurations)
GP 2.7 Identifioi ja ota mukaan asiaankuuluvat asianosaiset (Identify and Involve Relevant Stakeholders)
GP 2.8 Tarkkaile ja kontrolloi prosesseja (Monitor and Control the Process)
GP 2.9 Arvioi uskollisuus objektivisesti (Objectively Evaluate Adherence)
GP 2.10 Tarkastele tilannetta yhteistyössä ylemmän johdon kanssa (Review Status with Higher Level Management)

GG 3 Vakiinnuta määritellyt prosessit (Institutionalize a Defined Process)

Prosessi on vakiintunut määriteltynä prosessina.

GP 3.1 Luo määritelty prosessi (Establish a Defined Process)
GP 3.2 Kerää informaatiota kehityksestä (Collect Improvement Information)

GG 4 Vakiinnuta määrällisesti hallinnoitu prosessi (Institutionalize a Quantitatively Managed Process)

Prosessi on vakiintunut määrällisesti hallinnoituna prosessina.

GP 4.1 Luo määrällisiä tavoitteita prosessille (Establish Quantitative Objectives for the Process)
GP 4.2 Vakauta aliprosessin suorituskyky (Stabilize Subprocess Performance)

GG 5 Vakiinnuta optimointiprosessi (Institutionalize an Optimizing Process)
Prosessi on vakiintunut optimoivana prosessina

GP 5.1 Varmista jatkuva prosessin kehitys (Ensure Continuous Process Improvement)
GP 5.2 Korjaa ongelmien perimmäiset syyt (Correct Root Causes of Problems)