KYMENLAAKSON AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutusohjelma

Terhi Solanterä

PROJEKTINHALLINTAMENETELMIEN PM METHODOLOGY JA SCRUM YHDISTÄMINEN

Opinnäytetyö 2010

TIIVISTELMÄ

KYMENLAAKSON AMMATTIKORKEAKOULU
Tietojenkäsittely

SOLANTERÄ, TERHI	Projektinhallintamenetelmien PM Methodology ja Scrum yhdistäminen
Opinnäytetyö	47 sivua
Työn ohjaaja	Lehtori Päivi Hurri
Toimeksiantaja	Ohjelmapäällikkö Ralph Krasselt, Nokia Siemens Networks
Maaliskuu 2010	
Avainsanat	Projektinhallintamenetelmä, PM Methodology, Scrum, vesiputousmalli, ketterä menetelmä, hybridimenetelmä

Projektinhallintamenetelmä on joukko käytäntöjä, jotka antavat projektille sen tarvitsemat suuntaviivat. Perinteisenä pidettyä vesiputousmallia käytetään edelleen yleisesti myös ohjelmistokehitysprojekteissa, vaikka sille ominaiset pitkä ja huolellinen etukäteissuunnittelu, suhteellisen pitkä kehityssykli sekä joustamattomuus erityisesti muutosten suhteen voivat olla vaikeita sovittaa ohjelmistotekniikan nopeasti muuttuvaan maailmaan. Projekteilta vaaditaan yhä enemmän joustoa, tuloksia yhä lyhyemmällä aikataululla sekä nopeaa reagointia jatkuviin muutoksiin. Ketterät projektinhallintamenetelmät pystyvät vastaamaan näihin vaatimuksiin. Monet organisaatiot ovat kehittäneet myös ns. hybridimenetelmiä, joissa on pyritty yhdistelemään erityyppisten projektinhallintamenetelmien parhaina pidettyjä ominaisuuksia.

PM Methodology edustaa tyypillistä vesiputousmallia, ja Scrum on yksi tunnetuimpia ketteriä projektinhallintamenetelmiä. Nokia Siemens Networksissa päätettiin selvittää tulevia projekteja varten, voiko nämä kaksi erilaista metodologiaa yhdistää toimivaksi projektinhallintamenetelmäksi. Tämän opinnäytetyön tarkoituksena oli rakentaa PM Methodologysta ja Scrumista uudenlainen hybridimenetelmä.

Kirjallista aineistoa projektinhallinnasta sekä erilaisista menetelmistä on saatavilla runsaasti. Menetelmien yhdistelemisestä on myös jonkin verran materiaalia, mutta tietoa PM Methodologyn ja Scrumin yhdistämisestä ei löytynyt. Joissakin Nokia Siemens Networksin projekteissa on saatu jo kokemuksia näiden metodien yhdistämisestä. Tietoja aiemmista projekteista kerättiin haastattelemalla. Opinnäytetyön lopputuloksena rakentui uusi hybridimenetelmä, jota voi tarpeen mukaan käyttää ja soveltaa hyvin erilaisiin projekteihin.

ABSTRACT

KYMENLAAKSON AMMATTIKORKEAKOULU
University of Applied Sciences
Data Processing

SOLANTERÄ, TERHI	Combining Project Management Methodologies PM Methodology and Scrum
Bachelor's Thesis	47 pages
Supervisor	Päivi Hurri, Senior Lecturer
Commissioned by	Program Manager Ralph Krasselt, Nokia Siemens Networks
March 2010	
Keywords	Project management methodology, PM Methodology, Scurm, waterfall model, agile method, hybrid method

Project management methodology is a group of practices that give the project its guidelines. The traditional waterfall model is still widely used, even though the typically long and careful planning, relatively long development cycle and certain inflexibility especially concerning changes may be difficult to adapt to the quickly changing world of software engineering. More flexibility, results in a shorter schedule and quick reaction to constant changes are required from projects. The agile project management methodologies can respond to these requirements. Many organizations have also created so called hybrid methods in order to combine the best aspects of different methodologies.

PM Methodology represents a typical waterfall model and Scrum is one of the best known agile project management methodologies. It was decided in Nokia Siemens Networks to find out whether it would be possible to combine these two different methodologies and create a functional project management methodology to be used in future projects. The purpose of this thesis was to create a new hybrid method by combining PM Methodology and Scrum.

There is plenty of literature available about project management and different kinds of methodologies. Also some material about combining methodologies is available, but information about combining PM Methodology and Scrum could not be found. Some projects in Nokia Siemens Networks have already gained experience of combining these methods. Information from two different projects was gathered through interviews and emails. As a result of this thesis a new hybrid method was built. This method can be used and adjusted to very different kinds of projects.

SISÄLLYS

1 JOHDANTO

Projekteja voidaan sanoa olleen olemassa kaikilla elämän alueilla siitä lähtien, kuin ihmisillä on ollut mielessään suunnitelmia. Varhaisimmat esimerkit projektinhallinnasta juontavat juurensa jo esihistorialliselle ajalle, vaikka hankkeita ei olekaan osattu nimetä projekteiksi. (Litke & Kunow 2004, 7.) Projekti-sana on peräisin latinan kielestä ja tarkoittaa suunnitelmaa tai ehdotusta. Jokapäiväisiksi työmuodoiksi eri työpaikoilla projektit ovat tulleet vasta 20 – 30 vuotta sitten. (Rissanen 2002, 13-14.) Modernin projektinhallinnan varhaisimmat muodot liittyvät teolliseen vallankumoukseen ja suurten keksintöjen aikaan, joka lähti liikkeelle 1700-luvulla Iso-Britanniasta ja levisi nopeasti Ranskaan, Saksaan ja Yhdysvaltoihin (Murch 2002, 4).

Projekti voidaan määritellä kertaluonteiseksi tehtäväksi, jolla on sovitut tavoitteet, resurssit, organisaatio ja aikataulu (Haikala & Märijärvi 2004, 225). Suuret tehtäväkokonaisuudet voidaan jakaa myös pienempiin, peräkkäisiin tai rinnakkaisiin osaprojekteihin. Ohjelmistojen kehittämistyötä tehdään yleisimmin projektimuotoisesti. (Haikala & Märijärvi 2004, 53.) Metodologia tarkoittaa menetelmäoppia tai käytettyjen menetelmien kokonaisuutta. Se voidaan määritellä joukoksi toistettavia prosesseja eli käytäntöjä, joiden avulla pyritään tietynlaiseen lopputulokseen. (Murch 2002, 139.)

Projektinhallintaa voidaan sanoa laajaksi johtamiskonseptiksi, jonka tarkoituksena on saada hankkeita toteutettua sovitun ajan kuluessa, mahdollisimman edullisesti ja korkealuokkaisin lopputuloksin (Litke & Kunow 2004, 16). Projektinhallinnan perimmäisenä tarkoituksena on prosessien jatkuva parantaminen. Ihmisiä, prosesseja ja tekniikkaa yhdessä hyödyntämällä pyritään tekemään asioita yhä paremmin, nopeammin ja tehokkaammin. Työkalut ja menetelmät kehittyvät, mutta projektinhallinnan perusasiat pysyvät samoina. (Murch 2002, 10.)

Projektinhallintamenetelmä on joukko käytäntöjä, joka antaa projektille suuntaviivat, eli suosituksia ja neuvoja sekä menetelmät, eli tarkat prosessikuvaukset. Se sisältää yleensä myös projektinhallintatyökalut, sekä mallit dokumenteille ja tarkistuslistoille. Sen avulla erilaiset projektit voidaan toteuttaa samalla tavoin. Toisaalta erilaisten projektien toteuttamista varten tarvitaan myös erilaisia projektinhallintamenetelmiä. Valintaan vaikuttavat mm. projektin luonne ja laajuus, sekä sidosryhmien mielipiteet ja kokemukset.

Barry Boehm ja Richard Turner vakuuttavat kirjassaan ”Balancing Agility and Discipline: A Guide for the Perplexed”, että onnistuneeseen projektiin tarvitaan sekä suunnitelmapainotteista että ketterää menetelmää sopivasti yhdisteltynä. Menetelmien vastakkain asettaminen on turhaa, sillä kutakin projektia varten on mahdollista löytää sopiva tasapaino niiden välillä (Boehm & Turner 2008, 4 – 5). Useat organisaatiot ovat kehittäneet tällaisia ns. hybridimenetelmiä, joissa on yhdistelty erityyppisten projektinhallintamenetelmien parhaina pidettyjä ominaisuuksia. Myös eräissä Nokia Siemens Networksin (NSN) organisaatioissa on kerätty kokemuksia projekteista, joissa on yhdistetty suunnitelmapainotteinen Project Management Methodology (PM Methodology) for NSN BE and IT (Business Excellence and Information Technology) ja ketterä projektinhallintamenetelmä Scrum. Tämä on herättänyt kiinnostusta myös muiden projektien taholta yhä nopeammin muuttuvan ympäristön ja muuttuvien vaatimusten myötä.

2 HISTORIAA

”Kaikessa siinä, mitä pidetään tieteellisenä, pitää järjen olla hereillä ja ajatuksen toiminnassa. Sitä, joka tarkastelee maailmaa rationaalisesti, maailma katsoo rationaalisesti takaisin. Suhde on molemminpuolinen.” – Filosofi G.W.F. Hegel

Varhaisen projektinhallinnan merkittäviä pioneereja olivat amerikkalaiset Frederick Taylor (1856 – 1915), Henry Gantt (1861 – 1919) ja W. Edwards Deming (1900 – 1993) (Murch 2002, 5 – 9). Heidän johtamista ja projektinhallintaa uudistavat ajatuksensa ovat selkeästi parantaneet projektien tehokkuutta sekä laatua.

2.1 Liikkeenjohtamisen isä Frederick Taylor

Liikkeenjohtamisen isäksi mainitun Frederick Taylorin periaatteet saivat aikaan huomattavia tuotannonlisäyksiä. Hänen malliinsa sisältyivät mm. työtehtävien analysoiminen optimaalisten työtapojen määrittämiseksi, kouluttaminen tuottavampiin työskentelytapoihin, oikeiden taitojen ja ominaisuuksien tunnistaminen sekä yhdistäminen tehtävien ja tavoitteiden mukaisesti, kohtuullisen päivätyön standardin asettaminen tuottavuusodotuksia varten, työntekijöiden suorituskyvyn seuranta, palkitseminen kannustepalkkioin ja bonuksin, töiden johtaminen sekä raporttien laatiminen. (Murch 2002, 5.)

2.2 Gantt-kaavion isä Henry Gantt

Henry Gantt tunnetaan projektinhallinnassa edelleen käytössä olevasta Gantt-kaaviosta, joka on säilynyt pääosin samanlaisena sadan vuoden ajan. Gantt ilmaisi graafisessa muodossa projektinhallintaan liittyviä kysymyksiään, joita olivat esimerkiksi projektin kesto, kriittiset tehtävät, tehtävien vastuuhenkilöt, resurssit, viivästymisten ja mahdollisten muutosten vaikutukset, kokonaiskustannukset, tehtävien yksittäiskustannukset, projektin edistyminen, keinot korjata aikataulusta lipsuminen ja mahdolliset keinot nopeuttaa projektia. (Murch 2002, 6.)

2.3 Laatujohtamisen isä W. Edwards Deming

Laadun korostamiseen keskittynyttä W. Edwards Demingiä voidaan puolestaan kutsua laatujohtamisen isäksi. Hän ymmärsi, että laadun saavuttamiseksi kiintiöt ja virheistä rankaiseminen olivat vääriä johtamistekniikoita. Ajatuksiaan tukemaan hän loi 14 kohdan johtamisteorian, johon kuuluvat esimerkiksi kaikkien tuotteiden ja palvelujen laadun jatkuva parantaminen, uuden johtamisfilosofian omaksuminen, tuotanto- ja palvelujärjestelmän jatkuva parantaminen, työssä oppimisen edellytysten luominen, esimiesten johtajuuden kehittäminen, virheistä rankaisemisen lopettaminen, iskulauseiden, vaatimusten ja tulostavoitteiden poistaminen, numeeristen kiintiöiden poistaminen, esteiden poistaminen työylpeyden tuntemisen tieltä sekä jatkuvan opiskelun ja uudelleenkoulutuksen toimeenpaneminen. Projektien kokonaislaatu on parantunut selkeästi noudattamalla projektinhallinnassa Demingin oppeja. (Murch 2002, 7 – 9.)

3 PROJEKTINHALLINNAN METODOLOGIA

”*Jos et tiedä päämäärääsi, mikä tahansa polku vie sinut sinne.*” Vanha sioux-intiaanien sananlasku (Murch 2002, 139).

3.1 Mitä metodologia on?

Metodologia, eli metodiikka, tarkoittaa menetelmäoppia, tai myös käytettyjen menetelmien kokonaisuutta. Se voidaan määritellä joukoksi toistettavia prosesseja eli käytäntöjä, joilla pyritään tietynlaiseen lopputulokseen. Metodologian avulla erilaiset projektit voidaan toteuttaa samalla tavalla. Se antaa projektille suuntaviivat, eli toimintaan kohdistuvia suosituksia ja neuvoja, sekä menetelmät, eli tarkat prosessikuvaukset.

Metodologia sisältää yleensä myös projektinhallintatyökalut sekä mallit dokumenteille ja tarkistuslistoille. Metodologia luo standardit, joita noudattamalla voidaan saavuttaa merkittäviä etuja ja tuotannonlisäyksiä. (Murch 2002, 139.)

3.2 Metodologian käytön hyödyt

Taloudellisen perustelun eli business case -asiakirjan avulla määritellään esimerkiksi projektin budjetti, hyödyt, kustannukset sekä sitä varten tarvittavat resurssit. Projektia tukevat tekijät määritellään tarkasti mm. yritysjohtoa varten. Asiakirjaa päivitetään projektin aikana jatkuvasti, jotta tieto niin kustannuksiin kuin etuihinkin vaikuttavista muutoksista pysyy ajan tasalla. Yritysjohto ja asiakas ovat jo etukäteen selvillä projektin tavoitteista. Ennalta sovituissa hyväksymispisteissä heidän antamansa hyväksyntä varmistaa tavoitteiden toteutumisen. (Murch 2002, 140.)

Myös laatuvaatimukset ja niiden todentaminen, tarkistuslistat ja mallit ovat tärkeä osa hyvää metodologiaa. Laatukatselmusten ja -arviointien perusteella voidaan varmistaa, että ennalta sovittuja kehitysprosesseja on noudatettu, riskien tunnistaminen ja tiedottaminen on ollut hallinnassa ja projektin laajuus sekä tavoitteet ovat pysyneet sovituissa puitteissa. Erilaiset yllätykset kustannusten, aikataulun tai muiden riskien suhteen voidaan metodologioihin kuuluvien riskiarvioinnin ja -valvonnan avulla minimoida. Projektipäällikkö dokumentoi ja raportoi säännöllisesti yritysjohdolle projektin edistymisestä projektitiimin jäsenten tekemien ajankäyttöraporttien ja projektin työsuunnitelman perusteella. (Murch 2002, 141.)

Metodologioihin kuuluvien selvien toimintaohjeiden käytön on todettu lisäävän organisaatioiden tuottavuutta, koska näin voidaan paremmin keskittyä varsinaisen työn tekemiseen. Myös viestinnän on todettu parantuvan, sillä metodologian pohjalta määritellään tehtävät, aikataulu, tehtävien järjestys, suoritustavat, prosessien hallinta ja niistä tiedottaminen. (Murch 2002, 142.)

3.3 Ohjelmistokehityksessä käytettäviä projektinhallintamenetelmiä

Ohjelmistokehitysprojekteissa yleisesti käytettäviä projektinhallintamenetelmiä ovat mm. vesiputousmalli, prototyyppimalli, spiraalimalli, Rapid Applications-kehitys sekä ketterän ohjelmistokehityksen menetelmät (Agile Software Development), kuten Extreme Programming (XP), Adaptive Software Development (ASD) ja Scrum. Kaikille

menetelmille yhteisiä ovat tietyt käsitteet ja taktiikat. Niiden avulla yritetään ratkaista ja hallita samanlaisia organisaatioon ja projektinhallintaan liittyviä ongelmia. Jokaisella menetelmällä on omat vahvuutensa ja heikkoutensa. Oikeanlaisen, projektiin soveltuvan järjestelmän valitsemiseen tarvitaankin hyviä perustietoja ja riittävästi kokemusta. Sääntöjen ja niiden noudattamisen ei tulisi kuitenkaan rajoittaa yksilöiden myötävaikutusta ja ajattelua. Sääntöjä tulee voida myös tarkistaa ja parantaa. On myös olemassa tekijöitä, jotka voivat vaikuttaa esimerkiksi kaikkiin projektiaikatauluihin, joten valittu menetelmä ei ole koskaan ainoa syy siihen, jos aikataulu pettää. (Berkun 2006, 31 – 32).

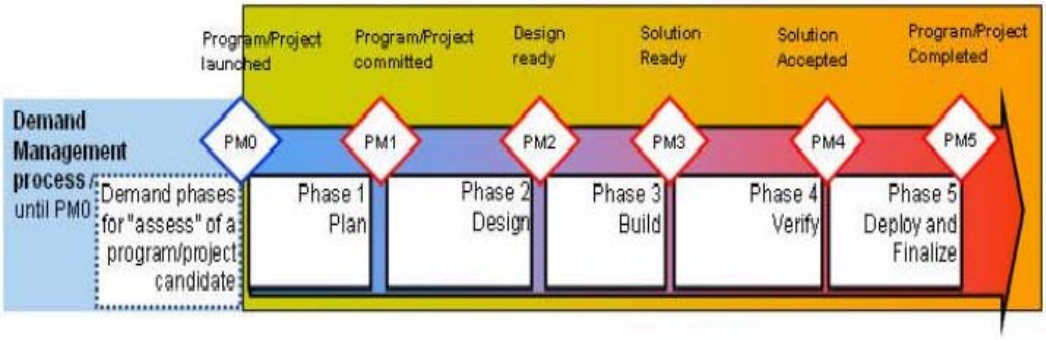
Perinteisenä pidetyssä vesiputousmallissa investoidaan ja varataan resursseja erityisen paljon suunnitteluun. Tehdyt päätökset ovat kaikkia sitovia mm. korkeiden muutoskustannusten vuoksi. Ketterät menetelmät puolestaan olettavat tulevaisuuden olevan täynnä muutoksia, ja ne perustuvatkin prosesseihin, joissa suunnitelmien muuttaminen on helppoa. Ohjelmistokehitysprojektit voivat usein olla jotakin näiden väliltä. Tällöin projektin alkuvaiheeseen sisältyy tarvekartoitusta ja suunnittelua, mutta työ jaetaan vaiheisiin, joissa kussakin on vielä omat suunnittelu-, toteutus- ja laadunvarmistusvaiheensa. Työn edetessä voidaan reagoida asiakkaan tarpeista tai muista syistä johtuviin muutoksiin. (Berkun 2006, 35 – 36.) Joissakin organisaatioissa on kehitteillä ns. hybridimenetelmiä, joissa on pyritty yhdistämään erilaisten menetelmien parhaat puolet.

4 PROJECT MANAGEMENT METHODOLOGY NOKIA SIEMENS NETWORKSISSÄ

4.1 Mikä on Project Management Methodology?

Project Management Methodology (PM Methodology) edustaa projektinhallintamenetelmänä varsin yleisesti käytössä olevaa, perinteistä vesiputousmallia. Se on käytössä mm. useissa Nokia Siemens Networksin (NSN) organisaatioissa. Project Management Methodology for NSN BE and IT (Business Excellence and Information Technology) on hyvin tyypillinen esimerkki vesiputousmallista. Se sisältää yksityiskohtaisia ohjeita ja valmiita malleja pakollisista sekä suositeltavista dokumenteista, joita tarvitaan erilaisten prosessien, ohjelmien ja projektien toteutuksessa. Sen avulla määritetään tarkasti esim. prosessien rajapinnat tai projektiin osallistuvien roolit ja vastuut niin, että ne sopivat yrityksen organisaatioiden toimintamalleihin. PM Methodologya noudatetaan uusien liiketoimintaprosessien, sovellusten ja IT-infrastruktuurin kehittämisessä, vanhojen muokkaamisessa ja täydentämisessä (uusien sovellusversioiden julkaisemi-

nen) sekä erilaisten konseptien ja tutkimusten toteuttamisessa. Menetelmää noudattamalla varmistetaan, että projektinhallinta täyttää tietyt vaatimukset, kuten SOX (Sarbanes-Oxley-lainsäädäntö) ja ISO900x (kansainväliset standardit). Kuvassa 1 on esitetty PM Methodologya noudattavan projektin eteneminen vaiheesta toiseen. (Project Management Methodology 2009.)



Kuva 1. Kaavakuva PM Methodologysta (Project Management Methodology 2009)

4.2 Projektin vaiheet PM Methodologyn mukaan

Projektin aloituspäätöstä edeltää aina määrittelyvaihe. PM Methodologyn mukaan itse projekti jakautuu viiteen vaiheeseen: projektin suunnitteluvaihe, järjestelmän suunnitteluvaihe, rakennusvaihe, varmennusvaihe sekä jakelu ja viimeistelyvaihe. Kukin vaihe päättyy merkkipaaluun, joita on myös kuusi: projektin käynnistäminen (PM0), projektin laajuuden, aikataulun ja resurssien lyöminen lukkoon (PM1), järjestelmän suunnittelun valmistuminen (PM2), ratkaisun valmistuminen (PM3), ratkaisun hyväksyminen (PM4) ja projektin päättäminen (PM5). (Project Management Methodology 2009.)

4.2.1 Projektin laajuuden ja rakenteen määritteleminen

Ennen varsinaista päätöstä projektin aloittamisesta käydään läpi vaatimusten käsittelyprosessi, jossa selvitetään ja määritellään asiakkaan ja liiketoiminnan edustajien tarpeet ja vaatimukset sekä niiden ratkaisumahdollisuudet mahdollista projektia varten. Projektin laajuus ja rakenne määritellään. Tässä vaiheessa tehdään laskelmia tarvittavista investoinneista ja työtehtävistä jo vaiheen 1 loppuun (PM1) saakka. Jos riittävät perusteet projektille löytyvät, tehdään päätös projektin aloittamisesta. Projektin esi-

vaihe päättyy merkkipaaluun PM0, jossa projekti varsinaisesti käynnistetään. (Project Management Methodology 2009.)

4.2.2 Vaihe 1: Projektin suunnittelu

Vaiheessa 1 laaditaan yksityiskohtaiset suunnitelmat. Projektin vaatimukset täydennetään, analysoidaan, määritellään tarkasti ja listataan. Projektin laajuus vahvistetaan, laaditaan tarkat laskelmat ja projektin arkkitehtuurista ja prosesseista sovitaan. Projektin työtehtävistä laaditaan suunnitelma. Mikäli vielä tässä vaiheessa ei voida laatia yksityiskohtaista työsuunnitelmaa aivan projektin loppuun asti, on vähimmäisvaatimuksena kuitenkin tarkan suunnitelman laatiminen merkkipaaluun PM2 asti, sekä karkeamman suunnitelman PM2:sta PM5:een. Investointilaskelmat täydennetään ja tarvittavista resursseista sovitaan. Vaihe 1 päättyy merkkipaaluun PM1, jolloin projektin laajuus, kustannukset, aikataulu, lopputulos ja resurssit lyödään lukkoon. (Project Management Methodology 2009.)

4.2.3 Vaihe 2: Järjestelmän suunnittelu

Toisessa vaiheessa keskitytään itse järjestelmän, eli projektikohteen suunnitteluun. Jos edellisessä vaiheessa on työtehtävistä laadittu tarkka suunnitelma vain PM2:een asti, se on tässä vaiheessa laadittava valmiiksi projektin loppuun saakka. Merkkipaalu PM2 on saavutettu, kun järjestelmän suunnitelma on valmis ja stabiloitu. (Project Management Methodology 2009.)

4.2.4 Vaihe 3: Rakentaminen

Kolmannessa vaiheessa tehdään varsinainen tekninen työ, eli rakennetaan itse projektin kohdetta, esimerkiksi uutta sovellusta. Osat rakennetaan, tarkistetaan ja yhdistetään kokonaisuudeksi. Systeemi ja prosessit muotoillaan, niitä testataan ja niistä tehdään simulointeja. Tässä vaiheessa testauksen suorittavat yleensä suunnittelijat ja arkkitehdit itse. Systeemin sekä prosessien hallintajärjestelmä ja tuki suunnitellaan tarkasti. Kun tarvittavat osat on integroitu kokonaisuuteen ja ratkaisun voidaan katsoa olevan valmis, on saavutettu PM3, ja ratkaisu voidaan luovuttaa tarkistettavaksi. (Project Management Methodology 2009.)

4.2.5 Vaihe 4: Verifioiminen

Neljännessä vaiheessa varmennetaan ratkaisun toiminnalliset ja ei-toiminnalliset ominaisuudet, sekä lopputulos kokonaisuudessaan. Tässä vaiheessa toteutetaan käyttäjätestaukset eli UAT (User Acceptance Testing). Systeemin sekä prosessien hallintajärjestelmä ja tukiorganisaatio perustetaan. Uuden järjestelmän julkaiseminen ja levitys suunnitellaan yksityiskohtaisesti. Merkkipaalu PM4 on saavutettu, kun ratkaisu on hyväksytty otettavaksi käyttöön. (Project Management Methodology 2009.)

4.2.6 Vaihe 5: Jakelu ja viimeistely

Viidennessä vaiheessa järjestelmä julkaistaan käyttöön. Loppukäyttäjille annetaan koulutusta ja mahdollisesti löytyviä virheitä korjataan. Järjestelmän hallinta- ja tukitehtävät luovutetaan niitä varten muodostetuille organisaatioille. Viimeinen vaihe loppuu merkkipaaluun PM5, kun julkaisemiseen ja jakeluun kuuluvat tehtävät on suoritettu ja tarvittavat dokumentit on viimeistelty, ja koko projekti päätetään. (Project Management Methodology 2009.)

4.3 PM Methodologyn kolme muotoa

Koska projektit voivat olla koon, sisällön, monimutkaisuuden ja riskien suhteen hyvin erilaisia, on NSN:ssä muokattu vesiputousmallista kolme hieman erilaista muotoa valittavaksi siten, että hallintamalli on samassa linjassa itse projektin kanssa. Nämä kolme versiota ovat Full Scale Mode (täysimittainen muoto), Medium Mode (keskikokoinen muoto) ja Light Mode (kevyt muoto). (Project Management Methodology 2009.)

4.3.1 Täysimittainen muoto

PM Methodologyn täysimittaista muotoa käytetään yleensä monimutkaisimmissa ja vaativimmissa projekteissa, kuten usein esim. kokonaan uuden sovelluksen tekemisessä. Tällöin projektin ohjausryhmän (Steering Group) hyväksyntä tarvitaan jokaisen vaiheen päätteeksi, ennen kuin voidaan siirtyä seuraavaan vaiheeseen. Täysimittaisessa muodossa toteutettavassa projektissa kaikki, tai lähes kaikki, projektinhallintaan sekä kustannushallintaan sisältyvät aktiviteetit, työkalut ja dokumentit ovat käytössä. (Project Management Methodology 2009.)

4.3.2 Keskikokoinen muoto

Keskikokoinen PM Methodologyn muoto sopii käytettäväksi hieman pienemmissä projekteissa, joissa kustannusten, riskien tai vaativuuden taso jää edellistä hieman alemmaksi. Esimerkiksi uuden sovellusversion kehittäminen voi olla tällainen projekti, jossa kustannukset ja tarvittavat resurssit ovat usein vähäisemmät, kuin kokonaan uutta sovellusta tehtäessä. Projektin ohjausryhmän hyväksyntä tarvitaan vain merkkipaalujen PM1, PM4 ja PM5 kohdalla. Tässä muodossa toteutettavassa projektissa noin puolet projektinhallintaan sekä kustannushallintaan liittyvistä aktiviteeteista, työkaluista ja dokumenteista on käytössä. (Project Management Methodology 2009.)

4.3.3 Kevyt muoto

Vaativuudeltaan tai kustannuksiltaan kaikkein pienimpiin projekteihin sopii käytettäväksi PM Methodologyn kevyin muoto. Tällainen projekti voi olla esimerkiksi jonkin sovelluksen pienimuotoinen kehittäminen tai vaikkapa vanhentuneen sovelluksen alasajo. Tällöin tarvitaan projektin ohjausryhmän hyväksyntä yleensä vain projektin alussa (PM1) ja lopussa (PM5). Vain vähimmäismäärä projektinhallintaan sekä kustannushallintaan liittyvistä aktiviteeteista, työkaluista ja dokumenteista on tällöin käytössä. (Project Management Methodology 2009.)

4.3.4 Sopivan projektimuodon valinta

Taulukosta 1 käyvät suuntaa-antavasti ilmi suositukset joiden mukaan voidaan valita projektille parhaiten soveltuva PM Methodologyn muoto. Kuten edellä on kerrottu, valinnassa tulee ottaa huomioon projektin riskitaso, vaativuus ja tarvittavan budjetin suuruus. Alkuperäisessä, NSN:n intranetissa olevassa taulukossa on budjetille asetettu selkeät euromääräiset rajat.

Taulukko 1. Projektimuodon valintamatriisi (Project Management Methodology 2009)

Riski/Vaativuus **Budjetti**	Alhainen	Keskitasoinen	Korkea
Alhainen	Kevyt	Kevyt	Keskikokoinen
Keskitasoinen	Keskikokoinen	Keskikokoinen	Täysimittainen
Korkea	Keskikokoinen	Täysimittainen	Täysimittainen

4.4 Tärkeimmät roolit ja niiden määritelmät

Projektin laajuus ja vaativuus vaikuttavat siihen, mitä kaikkia rooleja projektiin tarvitaan mukaan. Tärkeimpiä rooleja PM Methodologyn mukaisessa projektissa ovat ohjelmapäällikkö, projektipäällikkö, ohjausryhmä, arkkitehti, systeemin suunnittelija, prosessin kehittäjä ja palvelupäällikkö. (Project Management Methodology 2009.)

4.4.1 Ohjelmapäällikkö

Ohjelmapäällikkö (Program Manager) johtaa toisiinsa liittyvien projektien kokonaisuutta ja kunkin osaprojektin projektipäälliköt raportoivat hänelle. Ohjelmapäällikkö vastaa yhteistyössä projektipäälliköiden kanssa ohjelman kokonaissuunnitelmasta, vaatimusten priorisoinnista sekä osaprojektien ja koko ohjelman koordinoinnista yleensä. Hänen tehtävänään on myös hoitaa yhteydenpito ja yhteistyö muiden ohjelmien ja projektien kanssa. (Project Management Methodology 2009.)

4.4.2 Projektipäällikkö

Projektipäällikkö (Project Manager) toimii projektin vetäjänä ja on vastuussa projektin sovitusta sisällöstä, aikataulusta, kustannuksista, työn laadusta sekä projektitiimin johtamisesta. Hänen tulee seurata mm. projektin etenemistä tavoitteisiin nähden, kuluja, resurssien riittävyyttä, sekä laadun varmistamiseksi riskinhallintaa ja projektin dokumentointia. Projektipäällikkö osallistuu jo projektin alkusuunnitteluun ja vastaa projektin päättyessä valmiin tuotteen siirtämisestä käyttöön. (Project Management Methodology 2009.)

4.4.3 Ohjausryhmä

Ohjausryhmä (Steering Group) on ohjelman tai projektin ylin päättävä taho, joka on myös viime kädessä vastuussa koko ohjelmasta tai projektista. Sen tehtävänä on taata projektin onnistuminen varmistamalla, että lopputulos täyttää sille asetetut vaatimukset, sekä toimia osakkaiden ja muiden sidosryhmien edunvalvojana. Se ohjaa projektia antamalla selkeät ohjeet ja päätökset sekä toimii aktiivisesti ohjelmapäällikön ja projektipäällikön tukena. Ohjausryhmä kontrolloi ja hyväksyy projektin toimintakehykset, kuten projektin laajuuden, kustannukset ja aikataulun. Se kontrolloi projektin etenemistä hyväksymällä saavutetut merkkipaalut, seuraa raportointia ja kontrolloi myös

riskinhallintaa. Merkkipaalujen hyväksymiskokousten välillä pidetään säännöllisiä seurantakokouksia, joissa tarkastellaan paitsi projektin etenemistä ja tilannetta, myös mahdollisia muutosasioita. (Project Management Methodology 2009.)

4.4.4 Arkkitehti

Projektissa tulee olla mukana arkkitehti (Architect), joskus useampikin, erityisesti jos ollaan luomassa kokonaan uutta arkkitehtuuria ja ratkaisua, tai tekemässä suuria muutoksia olemassa olevaan arkkitehtuuriin. Arkkitehti osallistuu konseptin, tietojen ja sovellusten arkkitehtuurin suunnitteluun ja antaa ohjeita vaatimusten sekä sovellusten käsittelyyn. Hän tarkkailee arkkitehtuurin toteutusta ja valvoo standardien noudattamista, sekä pyrkii optimoimaan jo olemassa olevien, yhteiseen käyttöön tarkoitettujen elementtien käyttöä. (Project Management Methodology 2009.)

4.4.5 Systeemin suunnittelija ja prosessin kehittäjä

Systeemin suunnittelija (System Designer) vastaa sovelluksen rakenteesta ja toiminnoista. Kunkin ohjelmaversion suunnittelun ja kehityksen aikana hän johtaa suunnittelua, on mukana hyväksymisprosessissa sekä toimii kontaktihenkilönä arkkitehtuuriin ja konseptiin liittyvissä kysymyksissä. Hän myös vastaa alueeseensa kuuluvasta dokumentoinnista ja sen pysymisestä ajan tasalla. Prosessin kehittäjä (Process Developer) suunnittelee ja dokumentoi ohjelmaan liittyvät prosessit sekä tukee prosessipäällikköä (Process Manager) ja avainkäyttäjiä (Key User) mm. julkaisun ja levityksen aikana. (Project Management Methodology 2009.)

4.4.6 Palvelupäällikkö

Palvelupäällikkö (Service Manager) tarvitaan mukaan projekteihin, joissa muutetaan olemassa olevaa palvelua tai ollaan tekemässä kokonaan uutta palvelua. Hän vastaa olemassa olevan sovelluksen toiminnan jatkuvuudesta sekä ylläpitovaiheen toteutuksesta. Hän ottaa vastuun sovelluksesta projektipäälliköltä projektin päättyessä, kun sovellus otetaan käyttöön. Hän valvoo, että sovelluksen suorituskyky vastaa sovittua tasoa ja huolehtii mahdollisten vikojen korjauksesta sekä pienten muutosten toteuttamisesta. (Project Management Methodology 2009.)

4.4.7 Muita PM Methodology -rooleja

Projektin laajuudesta ja vaativuudesta riippuen on muitakin rooleja, joita saatetaan tarvita mukaan projektin toteuttamiseen. Osa rooleista voidaan yhdistää toisiinsa, kuten esim. pääinsinöörin (Chief Engineer) ja projektipäällikön roolit. Pääinsinöörin tehtävänä on lähinnä johtaa projektitiimin insinööri- ja systeemisuunnittelijaryhmää ja olla läheisessä yhteistyössä arkkitehdin kanssa. Muita rooleja ovat aluevastaava (Area Responsible), jonka vastuulla on tietty alue projektista sekä sitä hoitava ala-tiimi. Tehtävä voi myös olla yhdistetty projektipäällikön tehtäviin. Muutosjohtaja (Change Manager) tukee projektin hallintaa globaalisti sekä alueellisesti, on mukana levitysvaiheessa valmentamassa osakkaita ja muita sidosryhmiä sekä valvoo eri rooleihin ja vastuualueisiin liittyvää toteutusta, koulutusta ja tiedottamista. Käyttövaiheen valmistelupäällikkö (Use Phase Preparation Manager) on vastuussa kaikista käyttöönottoa valmistelevista toimenpiteistä ja ohjeistuksesta. Hän varmistaa osaltaan tuotteen toimituskelpoisuuden ottamalla huomioon kaikki käyttöönottovaiheeseen liittyvät näkökohdat. Tarkastajat (Reviewers) toimivat laaduntarkkailijoina. Heidän tehtävänsä on tarkistaa kaikki projektin tuotokset, ennen kuin niitä annetaan ohjausryhmän hyväksyttäväksi. PM Methodologyn mukaan tarvitaan kolmenlaisia tarkastajia: tietyn alueen ekspertit, arkkitehti ja laatujohtaja. (Project Management Methodology 2009.)

4.5 PM Methodologyyn sisältyvä dokumentaatio

PM Methodologyyn sisältyy mittava määrä dokumentaatiota. Aiheen laajuuden vuoksi se jätetään tässä työssä käsittelemättä tarkemmin. PM Methodologyyn liittyvien dokumenttien, ohjeiden ja mallien luettelo on jaettu 26 osaan projektin eri hallintoalueiden mukaan. Joihinkin osiin sisältyy vain yksi dokumentti ja siihen liittyvä ohje. Tällaisia ovat esimerkiksi liiketoimintariskin arviointi, projektin riskilista, sisäisen valvonnan arviointi, projektiehdotus, projektisuunnitelma ja projektin aikataulu. Kolmesta viimeksi mainitusta on tosin oma versionsa täysimittaiselle, keskikokoiselle ja kevyelle projektimuodolle. Useita kymmeniä dokumentteja sisältävistä osista mainittakoon esimerkiksi toimitettavien dokumenttien lista (Deliverables list), joka jakautuu neljään eri osioon: projektin hallinta, muutoksen hallinta, tekninen työ ja kehittäminen sekä käyttövaiheen valmistelu. Nämä osiot jakautuvat edelleen alaosioihin, joista löytyvät niihin kuuluvien dokumenttien mallipohjat ja ohjeet. (Project Management Methodology 2009.)

4.6 Vesiputousmallin käyttö ohjelmistokehityksessä

Suunnitteluun painottuvaa vesiputousmallia käytettiin alun perin erityisesti tekniikan alan työskentelyssä, kuten laajoissa konerakennusprojekteissa ja laitekehityksessä, joihin se soveltuikin hyvin. Menettelytavat ohjelmistokehityksessä olivat alkuun varsin yleisesti suorastaan kurittomat, minkä seurauksena aikataulun pettäminen, budjetin ylitys ja huono laatu eivät olleet lainkaan harvinaisia. Vesiputousmallia sovellettiin myös ohjelmistokehityksen tarpeisiin, mutta ohjelmien tekemiseen eivät välttämättä aina sopineet samat järjestelmät ja säännöt, kuin koneiden rakentamiseen. Ratkaistakseen ongelman Yhdysvaltain puolustusministeriö alkoi kehittää ohjeistusta ja dokumentteja, joiden avulla ohjelmistokehitysprosessiin alettiin soveltaa järjestelmällistä lähestymistapaa. Myös muutamat yritykset, kuten IBM, Hitachi ja Siemens, kehittivät samanlaisia standardeja omaan sisäiseen käyttöönsä. Ohjelmistokehityksen jatkuvan kasvun myötä kehitettiin lisää järjestelmällisiä prosesseja, ohjaustekniikoita ja menetelmiä ohjelmistokehityksen systemaattiseen toteuttamiseen. (Boehm & Turner 2008, 10.)

Koska vesiputousmallissa panostetaan erityisen paljon suunnitteluun, sen jälkeen tehdyt päätökset ovat kaikkia sitovia. Asiakas saa kustannusarvion jo projektin alussa. Malli on selkeä ja projektin eri vaiheet seuraavat toisiaan suoraviivaisesti. Tarkistuspisteet ja dokumentit on kiinnitetty tarkasti kunkin vaiheen rajapinnoille niin, että edellisen vaiheen loppudokumentti toimii syötteenä seuraavalle vaiheelle. (Pohjonen 2002, 40.) Prosessien aikana kertyvien tietojen tarkka dokumentoiminen varmistaa osaltaan, että esim. jonkun avainhenkilön menettäminen ei kaada projektia tai että tarvittaessa henkilöitä voi siirtää projektista tai alaprojektista toiseen ilman suurta koulutustarvetta (Boehm, Turner 2008, 12).

Käytännössä kuitenkin suuren ja monimutkaisen projektin kaikista aikatauluista ja lopputuloksista etukäteen sopiminen sekä työn toteuttaminen tarkalleen suunnitelman mukaisesti on yleensä mahdotonta. Erilaisiin kehitysprojekteihin sisältyy paljon epävarmuustekijöitä. Tietyn vaiheen suoritus paljastaa usein edellisessä vaiheessa tehtyjä virheitä, jolloin prosessissa on peruutettava takaisinpäin virheiden korjaamista varten. Vesiputousmallissa peruuttaminen on kallista ja monimutkaista, sillä todellisuudessa kehityshankkeiden vaiheet ovat harvoin toisistaan riippumattomia, minkä vuoksi saatetaan joutua uusimaan jopa kaikki edeltävät vaiheet. Mallin toisena ongelmana voi-

daan pitää sitä, että varsinaisia tuloksia päästään esittämään asiakkaalle vasta projektin loppuvaiheessa. (Pohjonen 2002, 40.)

Vesiputousmallia käytetään edelleen yleisesti myös ohjelmistokehitysprojekteissa. Sille ominaiset pitkä ja huolellinen etukäteissuunnittelu, suhteellisen pitkä kehityssykli ja joustamattomuus erityisesti muutosten suhteen ovat monen mielestä kuitenkin vaikeasti sovitettavissa ohjelmistotekniikan nopeasti muuttuvaan maailmaan (Boehm, Turner 2008, 16). Projekteilta vaaditaan yhä enemmän joustoa, tuloksia yhä lyhyemmällä aikataululla ja nopeaa reagointia jatkuviin muutoksiin. Ketterien projektinhallintamenetelmien kehittyminen on lähtenyt liikkeelle nopeista kehityskokeiluista ja koerakentamisesta sekä siitä ajatuksesta, että ohjelmointi on ennemminkin käsityötä, kuin teollinen prosessi (Boehm & Turner 2008, 16). Tuotannolliset projektit etenevät yleensä hyvin tehdyn suunnitelman mukaisesti, mutta erilaisiin kehitysprojekteihin sisältyy paljon epävarmuustekijöitä. Kehitysprojektille annettu tavoite antaa toiminnalle suuntaviivat, mutta voi olla vaikea tehdä tarkkaa projektisuunnitelmaa. Se tarkentuu vasta hankkeen edetessä. (Rissanen 2002, 48.)

5 SCRUM

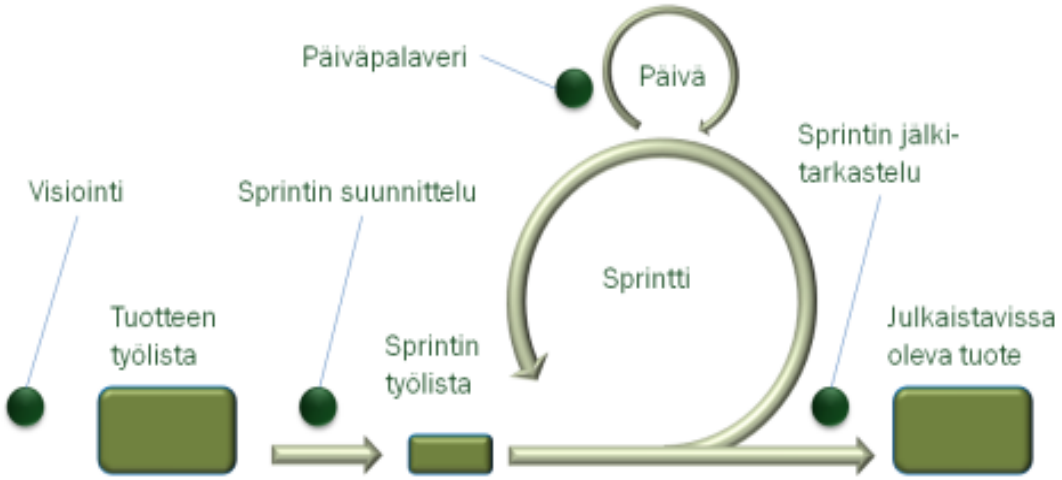
”*Minun mielestäni taistelu ei koskaan etene suunnitelmien mukaan. Suunnitelma on vain yhteinen perusta muutoksille. On erittäin tärkeää, että kaikki tuntevat suunnitelman, jotta sitä voidaan muuttaa helposti…nykyaikainen taistelu on erittäin liikkuvainen, ja päätökset on tehtävä nopeasti – pääsääntöisesti suunnitelmasta poiketen. Mutta kaikki ainakin tietävät, mistä ollaan tulossa ja sitten osapuilleen minne ollaan menossa.****”* - Kenraali Dan Laner, Israelin puolustusvoimien komentaja (Berkun 2006, 375.)

5.1 Ketterän ohjelmistokehityksen taustaa

Ketterän ohjelmistokehityksen juuret ovat 1970-luvun kehittämiseen ja muutoksiin painottuneessa ohjelmistokehityksessä. Ketterän ohjelmistokehityksen manifestissa vuodelta 2001 painotetaan mm. yksilöitä ja heidän välistään vuorovaikutusta enemmän kuin prosesseja ja työkaluja, toimivaa ohjelmistoa enemmän kuin kattavaa dokumentointia, yhteistyötä asiakkaan kanssa enemmän kuin sopimusneuvotteluja ja muutoksiin vastaamista enemmän kuin suunnitelmassa pysymistä (Agile Manifesto).

5.2 Mikä on Scrum?

Scrum-prosessin on kehittänyt vuonna 1993 amerikkalainen Jeff Sutherland yhdessä Ken Schwaberin kanssa. Termi ”scrum” on lainattu Rugby-pelistä. Sutherland sai ideaan nimestä luettuaan tutkimusta, jossa verrattiin huipputehokkaita, ristiin toimivia tiimejä Rugbyn aloitusmuodostelmaan. (What is Scrum?, ScrumAlliance.) Nykyisin Scrum on yksi yleisimmin käytetyistä ketterän ohjelmistokehityksen muodoista. Tässä projektinhallinnan mallissa ohjelmistokehitys rakentuu erimittaisten, iteratiivisten eli toistuvien syklien mukaan. Tärkeimmät syklit ovat sprintti ja päivä. Sprintin eli kehitysjakson tyypillisin kesto on 2 – 4 viikkoa, mutta voi joskus vaihdella viikosta jopa kahteen kuukauteen. Kuvassa 2 näkyy Scrumia noudattavan projektin eteneminen vaiheesta toiseen. (Scrum, Ketterät käytännöt.fi.)



Kuva 2. Kaavakuva Scrumista (Scrum, Ketterät käytännöt.fi)

5.3 Scrum-projektin roolit

5.3.1 Tiimi

Scrum-tiimiin ei kuulu perinteisiä ohjelmistokehitykseen liittyviä rooleja, kuten suunnittelija, arkkitehti, ohjelmoija, testaaja tai käyttöliittymäsuunnittelija. Tiimi on ristiin toimiva, itseohjautuva ryhmä, joka organisoi oman työnsä. Jäsenet valitsevat listalta itse päivittäiset tehtävänsä. Tiimi sitoutuu toimittamaan tietyt toiminnallisuudet asiakkaan käyttöön ja sen jäsenillä on kaikki siihen tarvittava asiantuntemus. Jokaisen sprintin lopputuloksena saadaan toimituskelpoinen tuote tai jokin lisäosa. Projektin

antamien suuntaviivojen puitteissa tiimi saa vapauden valita itse sopivimmaksi katsomansa työtavat sprintin tavoitteen saavuttamiseksi. Sprintin päättyessä työn lopputuloksesta esitetään demonstraatio tuotteen omistajalle. Tyypillinen koko Scrum-tiimille on 5 – 9 henkeä. (Scrum, Eclipse Process Framework Wiki.)

5.3.2 Scrum-mestari

Scrum-mestari (Scrum Master) jakaa tiimin kanssa vastuun töiden edistymisestä, mutta ei itse osallistu tiimin töihin. Hän varmistaa, että tiimi pysyy jatkuvasti toimivana ja tuottavana ja että se seuraa toimissaan Scrumin arvoja ja käytäntöjä. Tiimiläiset kertovat päiväpalavereissa ongelmista, jotka haittaavat työn etenemistä ja Scrum-mestarin tehtävänä on ratkoa ja poistaa nämä ongelmat. Hän ei ole projektipäällikkö, vaan Scrum-tiimin ohjaaja, yhteyshenkilö ja Scrum-prosessin johtaja. Hän vastaa päivittäisten Scrum-palaverien järjestämisestä. Käytännössä Scrum-mestarina toimii usein jonkin tiimin esimies tai projektipäällikkö. (Scrum, Eclipse Process Framework Wiki.)

5.3.3 Tuotteen omistaja

Tuotteen omistaja (Product Owner) edustaa osakkaita ja muita sidosryhmiä, valvoo heidän etujaan, päättää tuotteen ominaisuuksista ja niihin vaikuttavista seikoista sekä vastaa tuotteen kannattavuudesta. Hänen tehtävänään on listata vaaditut ominaisuudet ja priorisoida ne, eli laatia tuotteen työlista (Product Backlog). Hän päivittää listaa haluttujen ominaisuuksien sekä priorisoinnin osalta niin, että jokaisen sprintin alussa on käytettävissä ajan tasalla oleva lista, jolta tehtäviä poimitaan. Hän päättää tuoteversioiden julkaisupäivämäärästä ja sisällöstä sekä vastaa sprinttien suunnittelukokouksien järjestämisestä. Kokouksessa Scrum-tiimi päättää, millaisen määrän se pystyy tuotteen työlistasta seuraavan sprintin aikana toteuttamaan ja laatii sen perusteella sprintin työlistan (Sprint Backlog). Sprintin suunnittelun lisäksi tuotteen omistaja osallistuu myös sprintin jälkitarkasteluun. Hänen on lisäksi oltava aina Scrum-tiimin tavoitettavissa mahdollisten kysymysten varalta. Hän voi olla esim. tuotepäällikkö, markkinoinnin tai asiakkaan edustaja tai toimittajan tekninen projektipäällikkö. (Scrum, Eclipse Process Framework Wiki.)

5.4 Scrum-projektin aktiviteetit

Ennen kuin projekti aloitetaan, luodaan korkean tason visio projektiin kohdistuvista odotuksista; miksi se pitää toteuttaa ja mitä saadaan lopputuloksena. Tiimi, Scrum-mestari ja tuotteen omistaja suunnittelevat projektin suuret linjat tekemällä julkaisusuunnitelman (Release Plan). Projektin päämäärät määritellään ottaen huomioon sisäiset ja ulkoiset tarpeet, vaatimukset sekä riippuvuudet. Yksityiskohtaista suunnitelmaa ei tehdä, mutta esim. sprinttien lukumäärä ja kesto, arvio kunkin julkaisun tuomasta lisäarvosta, projektiin tarvittavien henkilöiden tai tiimien lukumäärä, julkaistavien versioiden lukumäärä ja julkaisupäivät suunnitellaan ja sovitaan. Suunnitelma tehdään tuotteen alustavan työlistan, arvioidun etenemisvauhdin sekä mahdollisten lukkoon lyötyjen päivämäärien perusteella, joista tuotteen omistaja on sopinut asiakkaan kanssa. Tuotteen omistaja priorisoi työlistan, jonka perusteella kunkin sprintin suunnittelu aloitetaan. (Scrum, Eclipse Process Framework Wiki.)

5.4.1 Sprintin suunnittelukokous

Sprintin suunnittelukokousta varten varataan yleensä yksi työpäivä ja siihen osallistuvat tiimin kaikki jäsenet, Scrum-mestari sekä tuotteen omistaja. Kokouksen ensimmäisen puoliskon aikana tiimi analysoi tuotteen omistajan priorisoiman työlistan ja tuotteen omistaja kuvailee tärkeimmiksi asettamansa ominaisuudet. Kokouksen jälkimmäinen puolisko käytetään sprintin suunnitteluun ja tiimin jäsenet esittävät tuotteen omistajalle kysymyksiä. Tämän jälkeen tiimi valitsee listalta joukon vaatimuksia, jotka katsoo pystyvänsä yhden sprintin aikana toteuttamaan ja ne siirretään tuotteen työlistalta sprintin työlistaan. Kokoukseen osallistujat määrittelevät yhdessä sprintin päämäärän (Sprint Goal) ja laativat lyhyen kuvauksen siitä, mitä sprintin aikana pyritään saavuttamaan. Tämän perusteella sprintin katselmoinnissa arvioidaan jälkeenpäin sen onnistumista. (Scrum, Eclipse Process Framework Wiki.)

5.4.2 Päiväpalaveri

Scrum-mestari järjestää päiväpalaverit, joissa jokaisen tiimiläisen on oltava läsnä. Kuka tahansa muukin projektiin tai sen sidosryhmiin kuuluva henkilö voi osallistua palaveriin, mutta vain kuuntelijana. Palaveri alkaa aina sovittuna aikana ja riippumatta tiimin koosta siihen varataan aikaa 15 minuuttia. Scrum-mestarin vasemmalla puolel-

laan oleva tiimin jäsen aloittaa raportoimisen tilanteestaan ja kierros jatkuu vastapäivään, kunnes kaikki ovat vastanneet kolmeen kysymykseen:

- Mitä olet tehnyt eilen?
- Mitä aiot tehdä tänään?
- Onko esteitä, jotka vaikeuttavat tavoitteeseesi pääsyä?

Päiväpalaverissa saa puhua vain yksi henkilö kerrallaan, eikä sen aikana käydä muuta keskustelua. Keskittymällä siihen, mitä kukin on tehnyt eilen ja aikoo tehdä tänään, tiimi pysyy tarkasti selvillä jo suoritetuista ja vielä suorittamatta olevista tehtävistä. Scrum-mestarilla ei ole puheoikeutta päiväpalaverissa, vaan hän kirjaa ylös tiimin jäsenten saavutukset, tavoitteet ja ongelmat. Sprintin työlista ja arviot jäljellä olevasta työstä päivitetään ja pidetään kaikkien sidosryhmien nähtävillä. Aikaa ei päiväpalaverissa käytetä ongelmien ratkomiseen, vaan mikäli tarvetta ilmenee, kuka tahansa tiimistä voi järjestää erillisen kokouksen päiväpalaverin jälkeen. (Scrum, Eclipse Process Framework Wiki.)

5.4.3 Sprintti

Sprintin eli työvaiheen aikana tiimi toteuttaa valitsemansa toiminnallisuudet. Tiimi organisoi itsensä ja työnsä haluamallaan tavalla ja voi käyttää siihen tarvittaessa myös ulkopuolista apua. Sprintin aikana sen työlistaa ei saa muuttaa. Jos käy ilmi, että sprintti ei esimerkiksi liiketoimintaympäristön odottamattomista muutoksista johtuen olekaan toteuttamiskelpoinen, Scrum-mestari voi keskeyttää sen ja aloittaa uuden sprintin suunnittelukokouksen. Toisaalta, mikäli näyttää siltä, ettei kaikkia sprintin tavoitteita pystytä toteuttamaan, voidaan tuotteen omistajan kanssa yrittää neuvotella joidenkin ominaisuuksien siirtämisestä seuraaviin sprintteihin. Silti jokaisen sprintin suunnittelu noudattaa aina tavanomaista menettelyä, kun päätetään mitä ominaisuuksia sprinttiin otetaan mukaan. Vastaavasti, jos tiimi näyttää pystyvän toteuttamaan enemmän, kuin mihin oli sprintin aikana sitoutunut, tuotteen omistajan kanssa voidaan neuvotella, mitä muita ominaisuuksia voitaisiin siirtää meneillään olevan sprintin aikana toteutettavaksi. (Scrum, Ketterät käytännöt.fi.)

5.4.4 Sprintin katselmointi

Jokainen sprintti päättyy katselmointiin, jolloin tiimi esittelee tuotteen omistajalle, asiakkaalle ja muille sidosryhmille sprintin aikana valmistuneet toiminnallisuudet. Kokous pidetään mahdollisimman epämuodollisena, eikä sen valmisteluun käytetä aikaa kahta tuntia enempää. Tavallisin tapa on demonstraation tekeminen tuotantoympäristöä vastaavassa testiympäristössä. Katselmoinnin aluksi esitellään tuotteen työlista sekä sprintin tavoitteet. Suurimman osan katselmukseen käytetystä ajasta vie uusien toiminnallisuuksien esittely ja sidosryhmien kysymyksiin vastaaminen. Kukin katselmoija esittää oman mielipiteensä sprintin onnistumisesta. Tuotteen omistaja neuvottelee tiimiläisten ja sidosryhmien kanssa tuotteen työlistaan tehtävistä muutoksista. Kaikki kertovat omat näkemyksensä tuotteeseen tarvittavista muutoksista sekä niiden tärkeysjärjestyksestä. (Scrum, Eclipse Process Framework Wiki.)

5.4.5 Sprintin jälkitarkastelu

Katselmoinnin jälkeen tiimi kokoontuu sprintin jälkitarkastelua varten. Siihen osallistuvat kaikki tiimin jäsenet, Scrum-mestari ja halutessaan myös tuotteen omistaja. Jälkitarkastelun aikana kukin osallistuja kertoo, mikä hänen mielestään edellisessä sprintissä meni hyvin, mikä asia toimi ja mikä ei, sekä esittää myös parannusehdotuksia seuraavaa sprinttiä varten. Yhtenä lähestymistapana käytetään sitä, että kukin tiimin jäsen kertoo, mitä hän haluaisi alkaa tehdä, minkä tekemisen hän haluaisi lopettaa ja minkä tekemistä jatkaa. Scrum-mestari kirjaa ylös osallistujien havainnot. Selkeät muutosehdotukset priorisoidaan ja kirjataan seuraavan sprintin työlistaan. Jälkitarkastelun avulla pyritään prosessin jatkuvaan parantamiseen. (Scrum, Ketterät Käytännöt.fi.)

5.5 Scrum-projektin työlistat

Scrum-projektissa käytetään tuotteen työlistaa, toteutusvaiheen työlistaa sekä joskus myös julkaisun työlistaa (Release Backlog). Tuotteen työlista sisältää kaikki ne vaatimukset, jotka tuotteelle on suunniteltu toteutettavaksi. Kun tuotteen omistaja on priorisoinut tuotteen työlistan, siitä siirretään julkaisun työlistaan seuraavaa julkaisua varten valitut ominaisuudet. Julkaisun työlistalta valitaan toteutusvaiheen työlistalle seuraavassa sprintissä toteutettavat ominaisuudet. Mikäli julkaisun työlistaa ei käytetä, seuraavassa sprintissä toteutettavat ominaisuudet valitaan toteutusvaiheen työlistalle

suoraan tuotteen työlistalta. Toteutusvaiheen työlistaa päivitetään joka päivä suoritettujen tehtävien osalta. Lisäksi tiimiläisten on pidettävä ajan tasalla arviota jäljellä olevista tehtävistä ja niihin käytettävästä ajasta. Yhteenveto arvioista esitetään graafisesti (Burndown Chart) ja sitä pidetään päivitetyn työlistan ohella kaikkien sidosryhmien nähtävillä. (Scrum, Eclipse Process Framework Wiki.)

5.6 Scrumin käyttö ohjelmistokehityksessä

Ketteristä projektinhallintamenetelmistä on saatu hyviä kokemuksia pienehköissä projekteissa (Boehm & Turner 2008, 20). Niissä painotetaan mm. yksilöitä ja heidän välistään vuorovaikutusta enemmän kuin prosesseja ja työkaluja, toimivaa ohjelmistoa enemmän kuin kattavaa dokumentointia, yhteistyötä asiakkaan kanssa enemmän kuin sopimusneuvotteluja ja muutoksiin vastaamista enemmän kuin suunnitelmassa pysymistä (Agile Manifesto). Projekteilta vaaditaan yhä enemmän joustoa, tuloksia yhä lyhyemmällä aikataululla ja nopeaa reagointia jatkuviin muutoksiin. Scrum kykenee vastaamaan näihin vaatimuksiin. Toisaalta Scrumista, kuten muistakin ketteristä projektinhallintamenetelmistä puuttuu pitkän tähtäimen suunnitelmallisuus, ja se voikin sellaisenaan olla suuremman järjestelmän toteuttamiseen riittämätön. Ongelmaksi voi nousta myös se, kuinka projektin kaikkien eri sidosryhmien asenteet ja toimintatavat saadaan muuttumaan joustavammiksi.

5.7 Ketterien ohjelmistokehitysmenetelmien käyttö suurissa organisaatioissa

Petri Kettunen Helsingin teknillisestä korkeakoulusta julkaisi syksyllä 2009 väitöskirjan ketterien ohjelmistokehitysmenetelmien käytöstä suurissa organisaatioissa. Hänen tutkimuksensa päätuloksena oli, että ketterät ohjelmistokehitysmallit pystyvät periaatteessa vastaamaan moniin tyypillisiin suurtenkin organisaatioiden avainongelmiin ja tuomaan niille sekä kustannus- että muita hyötytekijöitä. Useimmissa tapauksissa muutokset ja parannukset pelkässä ohjelmistotiimissä eivät ole riittäviä, vaan todellisen ketteryyskyvykkyyden luominen vaatii muutoksia koko yrityksen tasolla. Erillisten ketterien ohjelmistotiimien yhdistäminen laajempaan organisaatioyhteyteen vain ”alhaalta-ylös” tai ”sisältä-ulos” periaatteella voi olla ongelmallista. Jotta ketterää ohjelmistokehitystä voitaisiin hyödyntää täysimittaisesti, koko ohjelmistokehitystoimintaa pitäisi Kettusen mukaan tarkastella strategisesta liiketoimintanäkökulmasta ”ulkoa-sisään” periaatteella, yhtenä osana organisaation arvontuottosysteemiä. (Kettunen 2009, Tiivistelmä.)

5.7.1 Ketteryyden arvioimisperusteita

Ketteryyden mittaamiseen, kuten sen määrittelemiseenkään, ei ole olemassa yhtenäistä tapaa. Tyypillisiä arvioitavia ominaisuuksia ovat kuitenkin mm. reagoimisaika ja muutosten aiheuttamat taloudelliset kustannukset. Ketteryys ohjelmistotuotannossa vaatii investointeja, mutta toisaalta oikeat ja riittävät investoinnit maksavat itsensä pidemmällä aikavälillä tarkasteltuna takaisin ja tuovat lisäksi pysyviä ja selviä hyötyjä. Esimerkiksi vanhan tuotteen kohdalla teknisten ratkaisujen yksinkertaistaminen voi jossain vaiheessa aiheuttaa lisätyötä, mutta toisaalta alentaa tulevaisuudessa tehtävien muutosten aiheuttamia kustannuksia. (Kettunen 2009, 31.) Tuotekehitysprosessien nopeuttaminen on ollut paljon esillä viime vuosina. Tuotekehityksen nopeus yhdistetään usein joustavuuteen. Nopeat tuotekehityssyklit edesauttavat asiakkaan tai ympäristön muuttuviin vaatimuksiin mukautumista ja jatkuvasti kehittyvän teknologian omaksumista. Kun toimitusaika on riittävän lyhyt, kehitystyön aikana tehtävien muutosten tarve voi myös vähentyä. (Kettunen 2009, 17.)

5.7.2 Tulosten kerääminen, analysoiminen ja vertailu

Ketterän ohjelmistokehityksen pitkän aikavälin vaikutukset eivät monissa suurissa organisaatioissa ole vielä tarkkaan selvillä. Jotkut asiat vaativat myös kokonaan uudenlaista näkökulmaa ja arvioimistapaa, kuin perinteisinä pidettyjen ohjelmistokehitysmenetelmien arvioiminen. Oikeanlaisissa olosuhteissa ketterien ohjelmistokehitysmenetelmien käytöllä on kuitenkin saavutettu myönteisiä tuloksia. Useimmat raportoidut tulokset koskevat XP:tä tai Scrumia. Tyypillisimpiä saavutettuja hyötyjä ovat parempi markkinoille tulon ajoitus, tuottavuuden nousu, laadun paraneminen sekä mm. ohjelmistokehittäjien motivaation paraneminen. Toistaiseksi kuitenkin vain harvat suuret organisaatiot ovat julkaisseet tietoja ketterän ohjelmistokehityksen käyttöön liittyvistä tuloksista. (Kettunen 2009, 40.)

Kettusen tekemän systemaattisen vertailun tuloksena on, että vaikka ketterä ohjelmistokehitys vastaa moniin tyypillisiin tuotekehitysprojektien ongelmiin, saavutetut hyödyt eivät ole itsestään selviä, eivätkä ne välttämättä ole aina realisoitavissa ilman, että tehdään myös joitakin investointeja ja muutoksia. Tämän vuoksi on tärkeää tehdä tarkka kustannus- ja hyötyanalyysi. (Kettunen 2009, 71.)

Suurissa organisaatioissa eri projektien, tiimien ja toimintojen välillä on yleensä monimutkaisia sisäisiä riippuvuuksia, minkä vuoksi yhden erillisen tiimin pyrkimys noudattaa ketterää ohjelmistokehitystä ei todennäköisesti riitä tuomaan toivottua tulosta. Ketterästä ohjelmistokehityksestä kokonaisvaltaisen hyödyn saavuttaminen koko suuren organisaation tasolla edellyttää erillisten projektien ja tiimien tehokasta yhdentymistä. Ohjelmistokehitys on vain yksi osa koko yrityksen liiketoimintaa. Tavoitteet, ongelmat ja mahdolliset ratkaisut tulisi analysoida ottaen huomioon myös organisaation ympäristö. Kunkin tiimin asema, tehtävä ja merkitys organisaatiossa tulisi olla selvä. Ketteryyttä ja sen vaikutuksia tulisi tarkastella kokonaisvaltaisesti, koko yrityksen perspektiivistä.(Kettunen 2009, 76 - 88.)

6 MENETELMIEN VALINTA JA YHDISTÄMINEN KIRJALLISUUDESSA

6.1 Projektinhallinnan kehittämisen tarpeellisuus

Ketterien käytäntöjen hyväksyminen, omaksuminen ja edelleen kehittäminen vaatii kulttuurin muutosta koko organisaatiossa sekä projektin eri sidosryhmissä, sillä se vaikuttaa kaikkiin. Esimerkiksi pariohjelmointi vaatii kykyä oppia ja sopeutua läheiseen työskentely toisen kanssa. Myös organisaation johdon täytyy oppia ymmärtämään, että kaksi ihmistä yhdessä voivat olla tuottavampia ja mukautumiskykyisempiä, kuin kaksi erillään työskentelevää ihmistä. (Boehm & Turner 2008, 20.) Ohjelmistojen kehittämien vaatii nykymaailmassa jatkuvaa sopeutumista muutoksiin ja kykyä hallita suurta määrää erilaisia käyttöympäristöjä ja vaatimuksia (Boehm & Turner 2008, 23).

Ohjelmistoalalla on tavallista, että hyvin suuri osa projekteista tai kehityshankkeista epäonnistuu jollakin tavalla. Työ saattaa jäädä keskeneräiseksi, tai projektille varatut resurssit ylittyvät. Yleisimpänä syynä voidaan pitää projektien tai kehityshankkeiden hallinnassa olevia puutteita. Tietojärjestelmät kehittyvät yhä monimutkaisemmiksi ja laajemmiksi ja niille asetetut vaatimukset kasvavat jatkuvasti. Tuloksia vaaditaan yhä nopeammin. Yhä suuremmiksi kasvavat kehitysryhmät ovat työn organisoinnin ja hallinnan, sekä projektin sisäisen ja ulkoisen kommunikaation kannalta ongelmallisia. (Pohjonen 2002, 17.)

Organisaation ja sen toimintatapojen kehittäminen on pitkäjänteistä ja määrätietoista työtä, sillä sen tuomat hyödyt näkyvät vasta pidemmällä aikavälillä. Yleensä tarvitaan

enemmän kuin yksi projektikokonaisuus pidemmälle menevien kehityshankkeiden toteuttamiseksi ja kokemusten keräämiseksi. Prosessien kehittämisestä saatu hyöty voi kuitenkin hävitä, jos jokaisessa projektissa ryhdytään tekemään suuria muutoksia perusprosessimalliin. (Pohjonen 2002, 21 – 23.)

6.2 Sopivan tasapainon löytäminen perinteisen ja ketterän menetelmän välillä

Ei ole olemassa yksiselitteistä vastausta siihen, millainen projektinhallintamenetelmä olisi paras kullekin projektille. Erilaisia menetelmiä tarvitaan paitsi projektien erilaisuudesta johtuen, myös sen vuoksi, että projektihenkilöstöllä, organisaation johdolla, menetelmäarkkitehdillä tai asiakkaalla on omat näkemyksensä ja kokemuksensa asiasta. Tärkeimpiä huomioon otettavia seikkoja ovat tiimin koko ja jäsenten sijaintipaikat, projektin kriittisyys sekä sen tavoitteet. Menetelmästä riippumatta projektin tärkeimpänä päämääränä on pidettävä aina asiakkaan tarpeiden tyydyttämistä.

Kuten edellä on kerrottu, suunnitelmapainotteinen projektinhallintamenetelmä on hyvä vaihtoehto silloin, jos kyseessä on laaja ja monimutkainen systeemi, jonka turvallisuus ja luotettavuus ovat kriittisiä. Vaatimuksiin pitäisi myös olla odotettavissa muutoksia vain vähän ja ympäristötekijöiden tulisi olla jokseenkin helposti ennustettavissa. Ketterät menetelmät toimivat parhaiten silloin, kun systeemi ja kehitystiimi ovat pienempiä ja vaatimukset sekä ympäristö jatkuvasti muuttuvia. Asiakaan tulisi olla helposti tavoitettavissa ja myös halukas osallistumaan aktiivisesti projektiin. (Boehm & Turner 2008, 22.)

Monessa ajan suhteen kriittisessä projektissa onnistumisen kriittisin tekijä on kyky vastata nopeasti muutokseen. Toisaalta ylireagoiminen muutoksiin voi aiheuttaa huomattavia budjetin ylityksiä. (Boehm & Turner 2008, 22 – 23.) Lisäksi ketterille menetelmille tyypilliset mahdollisimman yksinkertaiset ratkaisut voivat myöhemmin johtaa kalliisti korjattavaan ongelmaan, joka tunnetaan ”arkkitehtuurin rikkojana.” Yksinkertainen suunnittelu alkuvaiheessa voi johtaa siihen, etteivät resurssit myöhemmin riitä systeemin suureen kasvattamiseen tai tarvittavien lisäominaisuuksien rakentamiseen. (Boehm & Turner 2008, 40 – 41.)

Suunnitelmapainotteiset menetelmät pohjautuvat yleensä asiakkaan ja ohjelmistokehittäjien väliseen sopimukseen, jossa on kyettävä ennakoimaan mahdolliset ongelmat ja dokumentoimaan tehtävät ratkaisut tarkasti. Vakaassa ympäristössä tämä on mo-

lempien osapuolien etu, sillä asiakas tietää, mitä tulee saamaan ja ohjelmistokehittäjät tietävät, mitä heiltä odotetaan. Toisaalta tällainen sopimus voi tehdä asiakassuhteesta suunnitelmapainotteisen menetelmän suurimman stressitekijän. Liian yksityiskohtainen sopimus voi esimerkiksi viivästyttää projektin aloittamista ja siihen on miltei mahdoton tehdä muutoksia. Liian epätarkka sopimus voi puolestaan aiheuttaa vääriä odotuksia ja johtaa ristiriitoihin sekä luottamuksen menetykseen. (Boehm & Turner 2008, 32.)

Suurten systeemien pitää löytää keinoja yhdistää käyttämiinsä menetelmiin ketteryyttä, jotta ne kykenevät säilyttämään asemansa ja vastaamaan asiakkaidensa odotuksiin. Ohjelmistojen koon ja monimutkaisuuden kasvaessa prosessien ja menetelmien pitää pystyä joustamaan. On myös tärkeää, että prosessit mitoitetaan projektin, tiimin ja ympäristön suhteen sopiviksi. Projektin johdon tulee kyetä tekemään oikeat johtopäätökset ottamalla huomioon tiimin koko, jäsenten ominaisuudet ja kokemus, projektin luonne, laajuus ja monimutkaisuus sekä budjetti, aikataulu, kriittisyys ja yleiset menettelytavat. (Boehm & Turner 2008, 23 – 24.)

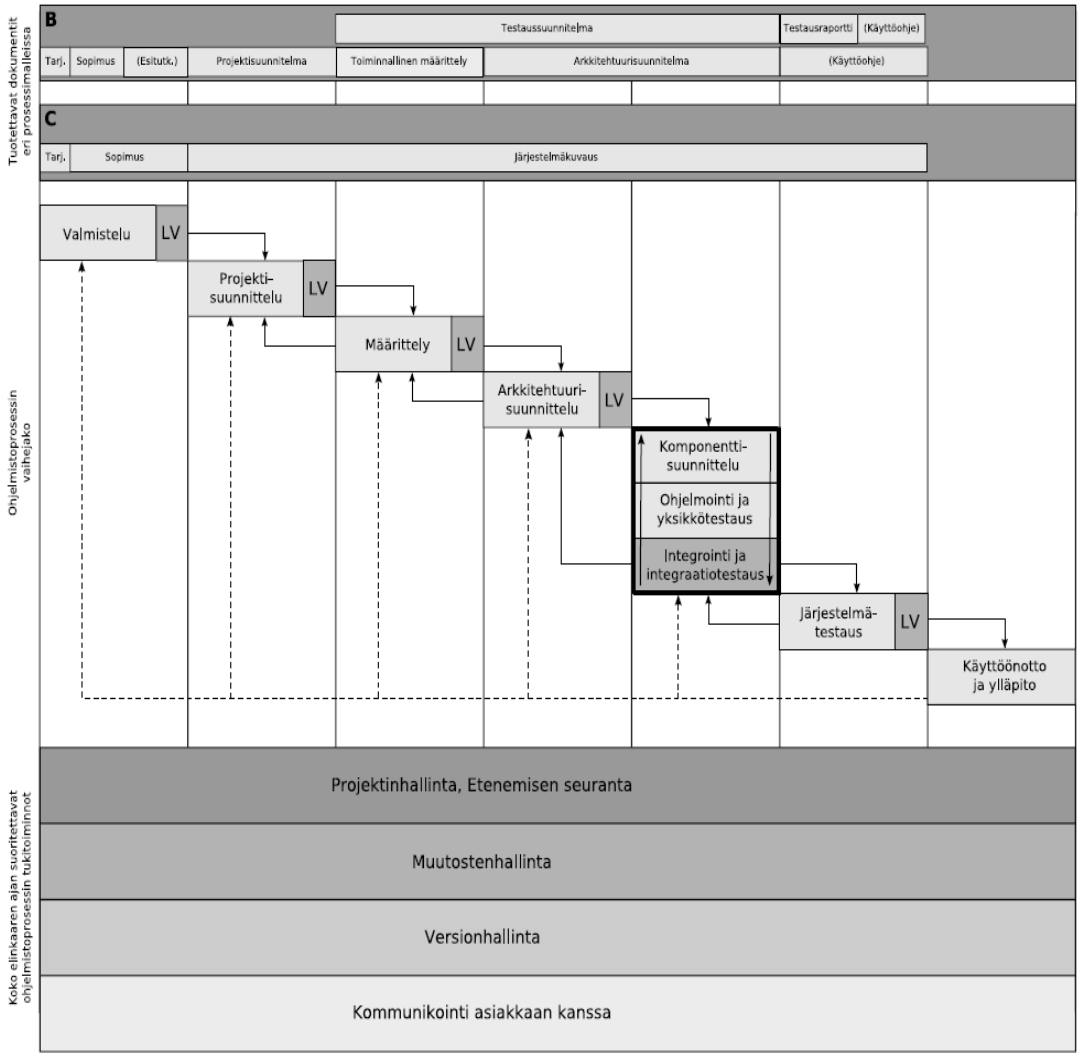
Riskien analysoiminen on tehokas keino mitoittaa prosessi kuhunkin projektiin sopivaksi. Riskejä voivat olla aikataulusta jälkeen jääminen, budjetin ylittyminen, tekninen epäonnistuminen tai voimavarojen odottamaton menettäminen. Riskianalyysin ydin, jolla oikeaa tasapainoa ketteryyden ja suunnitelmallisuuden välille haetaan, on yksinkertainen kysymys, jonka avulla analysoidaan prosessi kaikista eri näkökulmista: **”***Onko riskialttiimpaa käyttää (enemmän) tätä prosessin osaa vai välttää sen käyttämistä?***”** Vastauksen tarkka arvioiminen auttaa määrittelemään ja rakentamaan järkevän, tehokkaasti toimivan hybridimenetelmän, jossa ketteryys ja suunnitelmallisuus ovat oikeassa tasapainossa. (Boehm & Turner 2008, 24.)

6.3 Iteratiivinen suunnitelmaohjautuva prosessimalli

Mika Kuikka on vuonna 2008 tekemässään pro gradu -tutkielmassa kehittänyt aloittelevalle ohjelmistoyritykselle ohjelmistoprosessin, jossa yhdistyvät suunnitelmallisuus ja ketterälle ohjelmistokehitykselle tyypillinen iteratiivisuus. Kuvassa 3 näkyvät ohjelmistoprosessin elinkaaren lisäksi koko ajan rinnalla kulkevat tukitoiminnot, kuten projektin- ja muutostenhallinta. Mallissa on myös kuvattu kussakin vaiheessa tuotettava dokumentaatio. Jokaisen vaiheen jälkeen suoritetaan tuotetulle materiaalille laadunvarmistus (LV). Kaikki projektissa tuotettu materiaali viedään laadunvarmistuksen

jälkeen versionhallinnan alaisuuteen. Käännetyt ohjelmistot sekä osa dokumentaatiosta toimitetaan asiakkaalle, jonka hyväksyntä tarvitaan myös osaan dokumenteista. (Kuikka 2008, 61.)

Mallin perustavimpana osana on määrittelyvaihe. Määrittely vaatii huolellisuutta, koska jälkeenpäin tehdyt muutokset voivat koskea kaikkea tuotettua materiaalia ja tulla siten kalliiksi. Sen vuoksi määrittelydokumentti toimitetaan asiakkaalle kommentteja ja muutospyyntöjä varten ennen teknisen suunnittelun aloittamista. Tehtyjen muutosten jälkeen asiakas hyväksyy määrittelyn allekirjoittamalla dokumentin, mikä vähentää asiakkaan halua muuttaa vaatimuksia enää projektin aikana. (Kuikka 2008, 63.) Malli on lähempänä suunnitelmapainotteista kuin ketterää menetelmää, koska määrittelyvaiheeseen on panostettava paljon voimavaroja.



Kuva 3. Iteratiivinen suunnitelmaohjautuva prosessimalli (Kuikka 2008, 60)

7 UUDEN HYBRIDIMENETELMÄN KEHITTÄMISPROJEKTI

7.1 Projekti ”A” Nokia Siemens Networksissä

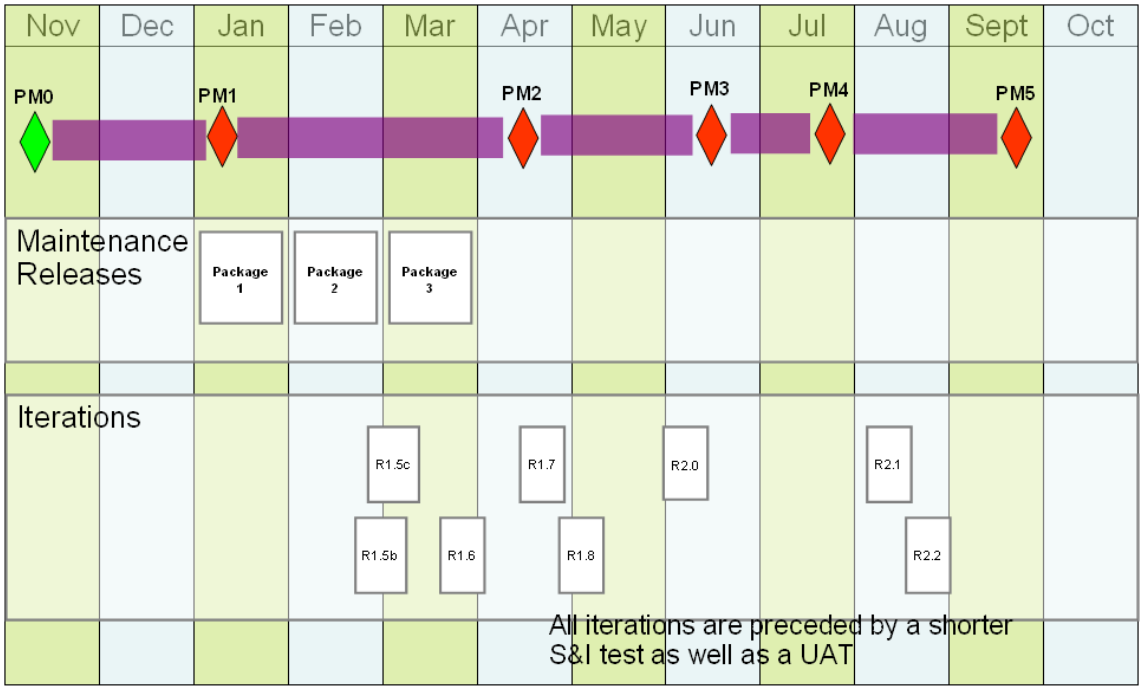
Nokia Siemens Networksissä käynnistettiin marraskuussa 2008 uusi projekti, jonka tarkoituksena oli jatkaa erään sovelluksen kehittämistä. NSN IT-organisaation kaikissa projekteissa käytetään projektinhallintamenetelmänä PM Methodologya, josta voi projektin vaativuuden mukaan valita täysimittaisen, keskikokoisen tai kevyen muodon. Edellä mainitussa projektissa päätettiin käyttää PM Methodologyn täysimittaista muotoa mm. sovelluksen kriittisyyden ja projektin laajuuden vuoksi. Sovellus oli vielä melko uusi ja projektin aikana oli tarkoitus toteuttaa monia uusia vaatimuksia. Suunnitelmiin sisältyi myös integraatioita eräiden muiden sovellusten kanssa.

7.1.1 Projektin aikataulu ja eteneminen

Projektin suunnitellusta kokonaiskestosta tuli noin 10 kuukautta ja sen sovittiin päättyvän syyskuussa 2009. Asiakkaiden käyttöön oli kuitenkin saatava uusia toimintoja jo paljon nopeammalla aikataululla, minkä vuoksi projektisuunnitelmaan otettiin mukaan myös vaikutteita ketteristä projektinhallintamenetelmistä. Projektisuunnitelmaan merkittiin projektin päätteeksi julkaistavan sovellusversion lisäksi julkaisupäivämäärät myös usealle väliversiolle. Ilman näitä välijulkaisuja asiakkaat olisivat joutuneet odottelemaan aina elokuun 2009 loppuun asti, ennen kuin olisivat saaneet suunnitellut uudet toiminnot käyttöönsä. (Krasselt, 10.3.2009.)

Merkkipaalu PM0:n ajankohta oli 21.11.2008, jolloin projektin operatiiviselta ohjausryhmältä tuli hyväksyntä projektin käynnistämiselle ja projektin suunnitteluvaihe voitiin aloittaa. PM1:n ajankohta projektisuunnitelmassa oli 19.2.2009, jolloin oli tarkoitus aloittaa itse järjestelmän suunnittelutyö. PM2:n suunniteltu ajankohta oli merkitty päivämäärälle 30.4.2009, jolloin järjestelmän suunnittelutyön odotettiin olevan valmiina ja varsinainen rakentaminen voitaisiin aloittaa. Merkkipaalu PM3:n suunniteltu ajankohta oli 30.6.2009, jolloin ratkaisun oli tarkoitus olla valmiina asennettavaksi ja testattavaksi. PM4 merkittiin projektisuunnitelmassa päivämäärälle 31.7.2009, mihin mennessä testauksien ja korjauksien odotettiin olevan valmiina ja asiakkaan hyväksyntä saatuna. PM5:n eli projektin päättymisen ajankohdaksi merkittiin 30.9.2009. (SCPR – Solution Cost and Planning Reporting 2009.)

Kuten edellä mainittiin, projektisuunnitelma poikkesi perinteisestä PM Methodologysta siten, että sovelluksesta oli merkitty PM1:n ja PM3:n välissä julkaistavaksi kaikkiaan 5 väliversiota, sekä PM4:n ja PM5:n välillä vielä 2 versiota. Kuvassa 4 näkyvät marraskuussa 2008 alkaneen projektin suunniteltu aikataulu ja eteneminen. Suunnitelmaa muutettiin vielä hieman kuvan laatimisen jälkeen, minkä vuoksi PM1 on merkitty kuvassa tammikuulle 2009. Uusien sovellusversioiden lisäksi suunnitelmaan oli merkitty myös tarpeelliset ylläpitojulkaisut. (Krasselt, 10.3.2009.)



Kuva 4. Projektin ”A” suunniteltu aikataulu ja eteneminen (Krasselt, 10.3.2009)

7.1.2 Projektista saatuja kehitysajatuksia

Projektin ohjelmapäällikkö halusi kehittää edelleen käytössä ollutta projektinhallintamenetelmää, sillä hänen mielestään välijulkaisujen sijoittaminen PM Methodologyn merkkipaalujen välille oli haasteellista. Ketteristä projektinhallintamenetelmistä häntä kiinnosti erityisesti Scrum, sillä hän oli kuullut eräistä muista Nokia Siemens Networksissä meneillään olevista projekteista, joiden projektinhallinnassa käytettiin Scrumia yhdistettynä PM Methodologyyn. Ongelmana oli, ettei vesiputousmallin ja Scrumin yhdistelmästä tuntunut löytyvän juurikaan tietoa. Valmiiden mallien etsiminen ei ollut tuottanut tulosta.

7.2 Opinnäyteprojekti

7.2.1 Sopimus opinnäyteprojektin tekemisestä

Ohjelmapäällikkö halusi selvittää, voisiko PM Methodologysta ja Scrumista kehittää sopivan yhdistelmämallin käytettäväksi sovelluksen tulevissa kehitysprojekteissa. Hänen johtamansa tiimin jäsen etsi alkuvuodesta 2009 sopivaa opinnäytetyön aihetta ja he sopivat tapaamisessaan 4.2.2009, että työntekijä perehtyisi projektinhallinta-aiheeseen ja voisi käyttää tehtävää opintoihinsa liittyvien seminaarityön, projektityön sekä opinnäytetyön aiheena.

Opinnäytetyön tavoitteeksi asetettiin, että työntekijä pystyisi kokoamiensa tietojen pohjalta rakentamaan oman ehdotuksensa tulevissa projekteissa käytettävästä yhdistelmämallista. Hänen tuli etsiä Scrumiin ja PM Methodologyyn, sekä erilaisten projektinhallintamenetelmien yhdistämiseen liittyvää materiaalia ja tutustua siihen. Lisäksi hänen piti haastatella jotakin asiantuntevaa henkilöä, kuten projektipäällikköä, kahdesta sellaisesta projektista, jossa oli käytetty Scrumin ja PM Methodologyn yhdistelmää.

7.2.2 Opinnäyteprojektin suunnitteleminen ja eteneminen

Opinnäyteprojekti suunniteltiin etenemään selkeästi kolmessa vaiheessa. Ensimmäisen vaiheen tavoitteena oli teoriatiedon kerääminen ja aiheeseen perehtyminen. Työntekijän tarkoituksena oli käyttää teoriaosuutta kevään 2009 seminaarityössä. Lisäksi sovittiin, että esimies saisi seminaarityöstä englanninkielisen käännöksen toukokuun 2009 loppuun mennessä. Seuraavan vaiheen tavoitteena oli käytännön kokemusten kerääminen haastattelujen avulla viimeistään kesäkuun 2009 loppuun mennessä. Viimeisen vaiheen tavoitteena oli johtopäätösten tekeminen ja uuden yhdistelmämallin rakentaminen kerättyjen tietojen pohjalta. Tavoitteena oli saada projektiraportti palautettua opettajalle viimeistään 1.9.2009 ja englanninkielinen käännös esimiehelle syyskuun 2009 loppuun mennessä. Opinnäytetyön tekemisestä laadittiin sopimus, jonka päättymispäiväksi merkittiin 31.3.2010.

Ohjelmapäällikkö antoi työntekijälle heti ensimmäisen tapaamisen päätteeksi omaa projektiaan koskevia tietoja ja materiaalia. He sopivat seuraavan puhelinpalaverin ajankohdaksi 10.3.2009, jotta työntekijällä olisi tarpeeksi aikaa hakea tietoja PM Methodologysta ja Scrumista sekä tutustua molempiin projektinhallintamenetelmiin. Tar-

koituksena oli, että hän olisi siihen mennessä perehtynyt aiheeseen riittävästi, jotta pystyisi esittämään kysymyksiä ohjelmapäällikölle sekä ottamaan myös kantaa käytössä olevaan projektinhallintamenetelmään.

Työntekijä etsi ensin yleistä tietoa vesiputousmallista ja ketteristä projektinhallintamenetelmistä. Sen jälkeen hän perehtyi tarkemmin PM Methodologyyn ja Scrumiin. Projektinhallintamenetelmistä oli saatavilla runsaasti tietoa alan kirjallisuudesta, Internetistä, sekä NSN:n Intranetistä. Erilaisten projektinhallintamenetelmien yhdistämisestä löytyi myös tietoa eri lähteistä, mutta valmiita malleja Scrumin ja vesiputousmallin yhdistelmistä ei löytynyt.

Maaliskuun 10. päivänä 2009 pidetyn puhelinpalaverin jälkeen alkoi tietojen kerääminen projekteista, joissa oli käytetty Scrumia yhdistettynä PM Methodologyyn. Projektipäälliköllä oli tiedossaan yksi tutkimuskohteeksi sopiva projekti, ja toisen sopivan projektin työntekijä löysi omien kontaktiensa kautta. Hän lähetti heti samana päivänä esimiehen kanssa pidetyn palaverin jälkeen kummankin projektin yhteyshenkilölle sähköpostiviestin, jossa esitteli asiansa ja pyysi lupaa haastattelun tekemiseen. Toisen projektin projektipäällikkö vastasi kysymyksiin sähköpostitse, toisen projektin ohjelmapäällikkö sopi työntekijän kanssa puhelinhaastatteluista. Ensimmäiset haastattelut tehtiin 20.3.2009 ja viimeiset 23.6.2009.

8 MUISTA PROJEKTEISTA KERÄTTYÄ TIETOA JA KOKEMUKSIA

”Niin kuin vanhaa rakennusta purettaessa tavallisesti varataan puretut rakennusaineet käytettäviksi uutta rakennettaessa, niin minäkin, hävittäessäni kaikki ne mielipiteeni, jotka katsoin löyhästi perustelluiksi, tein erilaisia havaintoja ja hankin itselleni monia kokemuksia, joista minulla on myöhemmin ollut hyötyä muodostaessani varmempia käsityksiä.” - Filosofi René Descartes

8.1 Projekti ”B” Nokia Siemens Networksissä

8.1.1 Taustatietoa projektista

Syksyllä 2008 NSN:ssä alettiin tarkistaa erästä sovellusta. Siitä löydettiin vakavia puutteita ja se päätettiin suunnitella ja rakentaa kokonaan uudella tavalla. IT-organisaation edustajat halusivat viedä projektin läpi joustavalla projektimenetelmällä,

mutta projektipäällikkö halusi kuitenkin mukaan PM Methodologyn, koska sitä noudattamalla projektin kunkin vaiheen valmistumista varten voitiin asettaa selkeät mittarit. Kyseessä oli suurehko projekti, johon suositusten mukaan olisi sopinut PM Methodologyn keskikokoinen muoto, mutta siinä päätettiin kuitenkin käyttää PM Methodologyn kevyttä muotoa yhdistettynä Scrumiin. (Vahter 18.3.2009.) Perusteluina PM Methodologyn kevyen muodon valitsemiselle oli, että koodaustyö oli ulkoistettu, projektitiimi oli suhteellisen pieni ja projektinhallintaa varten arvioitiin vähimmäismäärän dokumenteista riittävän (Vahter 20.3.2009).

8.1.2 Projektin aikataulu

PM0 ajoittui lokakuun 2008 alkuun, jolloin projektin suunnittelu käynnistyi. Merkkipaalu PM1 saavutettiin 18.12.2008, jolloin projektin aikataulu ja resurssit lyötiin lukkoon. Koska PM Methodologyn merkkipaalut toimivat kunkin vaiheen valmistuessa myös laskutuksen laukaisijoina, jätettiin PM2 (26.2.2009) ja PM3 (30.4.2009) projektisuunnitelmaan ”virtuaalisiksi” laskutuspäivämääriksi, mutta projektin etenemisen tarkistus- tai hyväksymispisteinä niitä ei käytetty. PM4:n päivämääräksi oli sovittu 15.5.2009 ja PM5:n päivämääräksi 30.5.2009. (Vahter 1.4.2009.)

8.1.3 Menetelmien yhteensovittaminen

Scrum sovitettiin projektiin mukaan siten, että varsinainen tekninen tuotanto jaettiin viiteen sprinttiin, jotka ajoitettiin PM1:n ja PM4:n välille. Sovellukseen rakennettavat elementit oli jaettu loogisiin kokonaisuuksiin, joiden mukaisesti kunkin sprintin pituus määräytyi. Ensimmäinen sprintti aloitettiin heti tammikuun 2009 alusta ja sen kestoksi oli sovittu 6 viikkoa. Toisen ja kolmannen sprintin kesto oli 4 viikkoa ja neljännen sekä viidennen kesto 3 viikkoa.

Projektipäällikkö ei puuttunut tuotantotiimin työskentelyyn, mutta valvoi, että kunkin sprintin tavoitteet saavutettiin. Osia joidenkin toimintojen toteuttamisesta siirrettiin myös seuraaviin sprintteihin, kun kävi ilmi, ettei jotakin ominaisuutta voitu saada meneillään olevan sprintin aikana valmiiksi. Vaiheet tuotteen kannalta katsottuna olivat joka sprintissä samanlaiset, kun sprintti tuotti uuden moduulin. Ensin ohjelmoijat testasivat koodin itse. Tämän jälkeen se asennettiin testipalvelimelle, jossa tuotteen omistaja tai joku avainkäyttäjä (Key User) testasi sen. Lopuksi valmis moduuli integroitiin

kokonaisuuteen. Projektissa ei tehty julkaisuja jokaisen sprintin lopussa, vaan täysin uusiksi rakennettu sovellus julkaistiin käyttöön vasta projektin päätteeksi.

Asiakkaan vaatimuksesta projektin päättymisajankohdasta tai budjetista ei voitu joustaa, vaan ne oli sovittu kiinteiksi. Mikäli projektin edetessä olisi käynyt ilmi, ettei kaikkia suunniteltuja ominaisuuksia olisi ehditty sovituissa puitteissa toteuttamaan, joustoa olisi tarvittaessa saatu jättämällä joitakin ominaisuuksia pois. Toisaalta myös asiakkaalla oli mahdollisuus tehdä muutoksia toivomuksiinsa vielä projektin edetessä. Joustoista neuvotteleminen vaati hyvää yhteistyötä ja neuvottelutaitoa tilaajan, tuottajan sekä ohjausryhmän välillä, kun alkuperäisten vaatimusten ja realistisesti toteutettavissa olevien moduulien väliltä oli löydettävä kompromissi. Tehtävä oli haasteellinen, mutta neuvottelujen tuloksena kukin osapuoli pyrki omalta osaltaan erityiseen joustavuuteen. Projektipäällikön mukaan projekti toimi hyvänä harjoituksena niin asiakkaalle, toimittajalle kuin ohjausryhmällekin. (Vahter 1.4.2009.)

8.1.4 Projektista saatu kokemus ja kehitysajatuksia

Projektipäällikön mielestä PM Methodologyn ja Scrumin yhdistäminen onnistui projektissa hyvin. Työ eteni sujuvasti ja moduulit valmistuivat suunnitellussa aikataulussa. Projektista saamansa kokemuksen perusteella hän oli sitä mieltä, että merkkipaalut PM2 ja PM3 voisi jättää kokonaan pois projektisuunnitelmasta, sillä hänen mielestään niiden sovittaminen yhteen ketterän projektinhallintamallin kanssa on haasteellista. Alun perin hän halusi itse PM Methodologyn mukaan projektiin, mutta Scrumista saadut kokemukset olivat hyviä ja hän sanoi toivovansa, että kehitystyössä voitaisiin noudattaa tästä eteenpäin vielä puhtaammin joustavaa projektinhallintamallia. (Vahter 20.4.2009.)

8.2 Projekti ”C” Nokia Siemens Networksissä

8.2.1 Taustatietoa projektista

Vuoden 2008 kesällä Nokia Siemens Networksissä aloitettiin erään sovelluksen kohdalla uusi projekti, jonka tavoitteena oli sovelluksen integroiminen toisen sovelluksen kanssa. Projekti haluttiin viedä läpi mahdollisimman joustavasti ja siinä päätettiin käyttää PM Methodologyn kevyttä muotoa yhdistettynä Scrumiin. Samaan aikaan oli käynnissä myös useita rinnakkaisprojekteja, sillä sovellus piti integroida muutaman

muunkin sovelluksen kanssa. Tämä oli projektin aikataulun suunnittelemisen ja noudattamisen kannalta erityisen haasteellista.

8.2.2 Projektin aikataulu

Projektin aloituspäivä oli 18.6.2008 ja suunniteltu päätösajankohta toukokuun 2009 loppu. Merkkipaalut PM0 ja PM1 yhdistettiin, samoin kuin merkkipaalut PM4 ja PM5. Projektista jätettiin pois merkkipaalut PM2 ja PM3. Ohjausryhmän hyväksyntä tarvittiin vain projektin alussa ja lopussa. (Pietarila 6.4.2009.) Myöhemmin projektin päätösajankohtaa jouduttiin siirtämään useaan otteeseen, kunnes projekti saatiin lopulta päätökseen marraskuussa 2009.

8.2.3 Menetelmien yhteensovittaminen

Työt sovittiin toteutettavaksi PM1:n ja PM4:n välillä 12:ssa kuukauden mittaisessa sprintissä. Arkkitehti, tuotteen omistaja ja projektipäällikkö yhdessä suunnittelivat tuotteen työlistan (Product Backlog) ja sprinttien määrän sekä keston. Resursointi tehtiin siten, että kaikki tuotteen työlistaan otetut pakollisiksi luokitellut vaatimukset pystyttäisiin projektin aikana toteuttamaan. Niiden lisäksi piti pystyä toteuttamaan myös osa niistä työlistaan otetuista vaatimuksista, joita ei ollut luokiteltu pakollisiksi, mutta kuitenkin tarpeellisiksi. Luokittelussa otettiin huomioon myös asiakkaan painotukset eri vaatimusten tärkeydestä. Tuotteen omistaja vastasi tuotteen sekä projektin työlistan priorisoimisesta. Scrum-tiimi puolestaan oli itse vastuussa sprinttien työlistoista, joita ei voinut sprinttien aikana enää muuttaa. (Tusa 6.4.2009.)

Myös projektin ohjausryhmän toiminta erosi perinteisestä mallista. Vaatimusten määrittelyssä ennen ohjausryhmän hyväksymää PM1:ä ei menty tarkkoihin yksityiskohtiin, vaan ominaisuudet määriteltiin korkeammalla tasolla. Vaikka ohjausryhmän hyväksyntä tarvittiin vain projektin alussa ja lopussa, ryhmä seurasi projektia päivittäin, sillä tuotteen omistajat kuuluivat myös ohjausryhmään ja he olivat jatkuvasti mukana projektissa. Tällä oli selvä vaikutus ohjausryhmän joustavuuteen ja tilanteen tasalla pysymiseen. Projektin uudenlainen hallinta vaikutti sen kaikkiin sidosryhmiin, mutta ohjelmapäällikkö piti olennaisena sitä, että ohjausryhmä oppi ymmärtämään ja hyväksymään muuttuneet menetelmät ja työtavat. (Tusa 6.4.2009.)

8.2.4 Projektista saatu kokemus ja kehitysajatuksia

PM Methodologyn ja Scrumin yhteensovittaminen onnistui projektissa hyvin. Projektin ohjausryhmän ja asiakkaan kanssa sovittiin budjettiraamista sekä tietyistä aikarajoista, jolloin määrättyjen asioiden piti olla valmiina. Sovituissa puitteissa pystyttiinkin melko pitkään toimimaan, mutta myöhemmin tuli jonkin verran ongelmia budjetin, sekä erityisesti aikataulun suhteen. Projektin päättymistä jouduttiin siirtämään useaan otteeseen. Ohjelmapäällikön mukaan ongelmat eivät liittyneet kuitenkaan projektinhallintamenetelmään, vaan tilanne olisi voinut olla aivan sama käytetystä menetelmästä riippumatta. Integraatioasioissa ongelmat voivat johtua myös toisesta sovelluksesta, tai siitä että toinen osapuoli noudattaa aina tiukasti oma prosessiaan. (Tusa 26.3.2009.)

Projektista saamansa kokemuksen perusteella ohjelmapäällikkö oli sitä mieltä, että hankkeen suunnittelu, määrittelytyö ja erityisesti sisällön konseptointi ennen jokaista sprinttiä pitäisi tehdä hieman tarkemmin. Myös erilaisten riippuvuuksien tunnistaminen pitäisi pystyä tekemään paremmin, jotta ne pystyttäisiin poistamaan. Toisin sanoen ajatus ”ennen kuin teet tämän, nämä on oltava tehtynä” pitäisi olla selvä kaiken aikaa. Lisäksi kommunikaatio oman projektin ja ulkopuolisten tahojen välillä pitäisi saada toimimaan vielä tehokkaammin, vaikka se Scrumin avulla huomattavasti parantuikin perinteisellä tavalla toteutettuihin projekteihin verrattuna. Tulevissa projekteissa ohjelmapäällikkö lisäisi vielä ohjelmointivaiheen testausta, sekä testausten automatisointia (TDD = Test Driven Development). (Tusa 23.6.2009.)

9 SCRUMIN JA PM METHODOLOGYN YHDISTELMÄ PROJEKTILLE ”A”

”Jos joku uskoo löytäneensä elämän ongelman ratkaisun ja haluaisi sanoa itselleen, että nyt kaikki on aivan helppoa, hänen tarvitsisi vain muistaa - huomatakseen olevansa väärässä - että oli aika, jolloin tätä ratkaisua ei ollut löydetty, mutta siihenkin aikaan oli pystyttävä elämään, ja tämän nähdessään hän näkisi ratkaisunsa sattumanvaraisena.” – Filosofi Ludwig Wittgenstein

9.1 Uuden hybridimallin peruspiirteet

Projektinhallintamenetelmä, jossa yhdistetään Scrum ja PM Methodology, on rakennettu osittain projektin ”A” käyttämän projektisuunnitelman pohjalta. Kyseisen sovel-

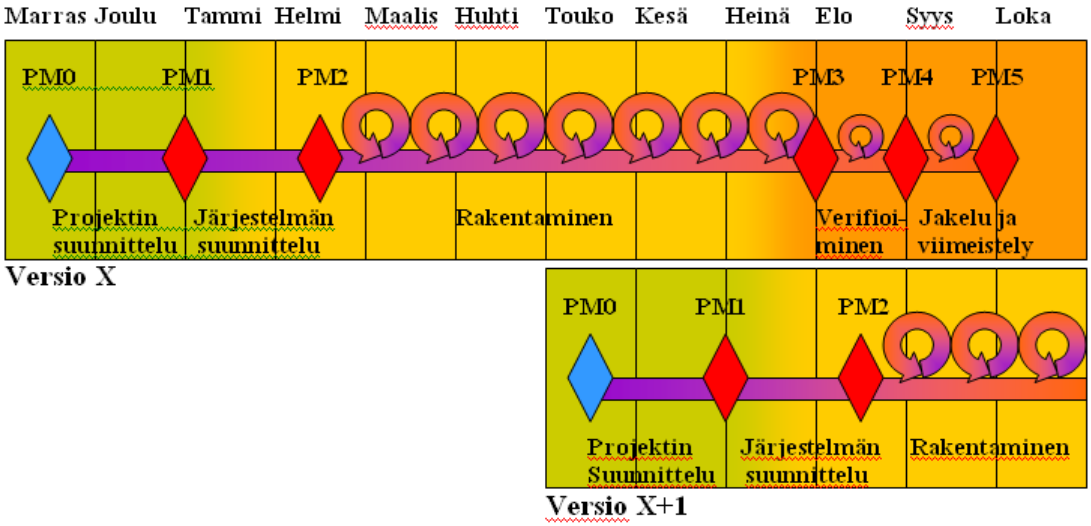
luksen kehitysprojekteissa on käytetty noin vuoden pituista sykliä. Yhdistelmämalli on kuvattu noudattamaan samaa kiertoa, jotta sitä on helpompi vertailla alkuperäiseen malliin. Sen mukaan menetelmän kehyksenä on PM Methodologyn täysimittainen muoto. Projekti etenee merkkipaalulta toiselle, mutta varsinainen ohjelmointityö suunnitellaan ja toteutetaan noudattamalla Scrum-prosessia. Mallia voidaan soveltaa myös PM Methodologyn keskikokoiseen tai kevyeen versioon projektien ”B” ja ”C” tapaan, jolloin projektiin saadaan mukaan enemmän joustavuutta.

9.1.1 Scrum ja PM Methdologyn täysimittainen muoto

Scrumin ja täysimittaisen PM Methodologyn yhdistelmämalli on esitetty kuvassa 5, josta näkyvät projektin vaiheet ja eteneminen, sekä sprinttien sijoittuminen PM Methodologyn merkkipaaluihin nähden. PM Methodologyn täysimittaista muotoa noudattava yhdistelmämalli eroaa joustavimmillaan perinteisestä vesiputousmallista eniten siinä, että konseptia ei varsinaisesti jäädytetä PM2:ssa, vaan vasta ennen jokaista sprinttiä. Tämä edellyttää sitä, että aikataulun, kustannusten, toteutettavien ominaisuuksien ja resurssien osalta riittää korkeamman tason määrittely ilman kaikkein tarkimpia yksityiskohtia. Aikataulun ja kustannusten joustoa varten voidaan neuvotella esimerkiksi minimi-, tavoite- ja maksimiarvoilla määritelty kehys, jonka sisällä toimitaan. Tarkoituksena on, että suunnittelu-, toteutus- ja testaustyöt jatkuvat jokaisessa sprintissä ja kukin sprintti päättyy tuotoksen julkaisuun. Verifioimisvaihe on tällöin hyvin lyhyt, sillä varsinaiset testaukset suoritetaan aina ennen kunkin sprintin päättymistä. Mikäli konseptin jäädyttämisestä PM2:ssa ei pystytä joustamaan, voidaan varsinainen rakentaminen aloittaa joustavasti jo järjestelmän suunnitteluvaiheen aikana, jolloin vähintään yksi sprintti toteutetaan ennen konseptin jäädyttämistä. Kun joitakin tärkeimpiä ominaisuuksia on saatu toteutettua ennen PM2:ta, voidaan mahdollisesti esiin tulleet muutostarpeet ottaa vielä konseptin laatimisessa huomioon. Tällaisena projekti noudattaa kuitenkin lähinnä perinteistä vesiputousmallia, kuten kuvasta 5 käy ilmi.

Toteutettavia ominaisuuksia varten laaditaan tuotteen työlista, johon priorisoidaan vaatimukset, jotka on pakko toteuttaa projektin aikana. Listaan sisältyy aina myös joitakin ei-pakollisia vaatimuksia, joista osa pystytään sovittujen resurssien puitteissa toteuttamaan. Tämä mahdollistaa osaltaan myös jouston niissä tilanteissa, jolloin asiakkaalta tai joltain muulta taholta tulee kiireellinen muutosvaatimus. Muutosvaatimukset

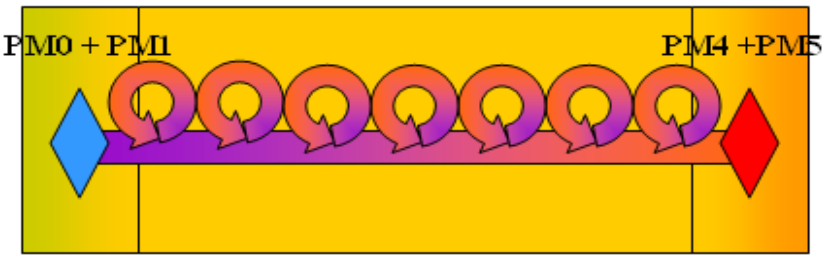
käsitellään normaalisti omassa prosessissaan ja lisätään hyväksymisen jälkeen seuraavan sprintin työlistaan. Kiireellisissä tapauksissa voidaan joustaa ja tehdä yksittäinen muutos tai korjaus poikkeuksellisesti nopeammalla aikataululla.



Kuva 5. Scrumin ja PM Methodologyn täysimittaisen muodon yhdistelmä

9.1.2 Scrum ja PM Methodologyn keskikokoinen tai kevyt muoto

Ketteryyttä yhdistelmämalliin saadaan, mikäli projektissa on mahdollista yhdistellä ja jättää pois PM Methodologyn merkkipaaluja, kuten projekteissa ”B” ja ”C” tehtiin. Tällöin projektiin sovelletaan PM Methodologyn keskikokoista tai kevyttä muotoa. Joustavimmillaan malli on silloin, kun merkkipaalut PM0 ja PM1, sekä PM4 ja PM5 yhdistetään ja PM2 sekä PM3 jätetään projektista pois. Kun ohjausryhmän hyväksyntä on saatu ja projekti aloitetaan, se etenee Scrum-prosessin mukaisesti sprinteissä. Tuotteen työlistan pohjalta suunnitellaan kunkin sprintin työlista, minkä mukaan suunnittelu-, toteutus- ja testaustyöt sprintin aikana tehdään. Jokaisen sprintin päätteeksi julkaistaan tuotos. Kun tuotteen työlistalta on saatu toteutettua riittävä määrä ominaisuuksia ja ohjausryhmältä sekä asiakkaalta on saatu hyväksyntä, projekti päätetään. Kuvassa 6 esitetään projektin eteneminen, kun käytetään PM Methodologyn kevyttä muotoa yhdistettynä Scrumiin.



Kuva 6. Scrumin ja PM Methodologyn kevyen muodon yhdistelmä

9.2 Projektin vaiheet ja eteneminen

Kuten aiemmin mainittiin, yhdistelmämalli on kuvattu noudattamaan samanlaista kiertoa kuin projekti ”A”. Uusi projekti lähtee liikkeelle edellisen projektin puolivälissä. Tämä mahdollistaa uusien ohjelmapakettien toimittamisen asiakkaalle ilman keskeytyksiä. Myös resurssit pysyvät tehokkaassa käytössä, sillä toteutusvaiheen päättyessä (PM3) osa resursseista vapautuu uuden projektin käyttöön. Verifiointi- ja viimeistelyvaiheessa ei enää tarvita kaikkien ohjelmoijien työpanosta, sillä silloin tehdään ainoastaan korjauksia käyttäjätestauksen (User Acceptance Testing) ja käyttöönoton yhteydessä mahdollisesti löytyneisiin virheisiin. Jos jokaisen sprintin jälkeen on tehty julkaisu ja kunkin ominaisuuden testaaminen on suoritettu jo sprintin aikana, verifiointivaiheessa riittää hyvin pienimuotoinen testaus. Mikäli tässä vaiheessa esitetään muutosvaatimuksia, ne ohjataan tavanomaisen käsittelyn ja hyväksymisen jälkeen seuraavan projektin työlistoihin.

9.3 Asiakkaan ja ohjausryhmän hyväksyntä uusille toimintatavoille

Asiakkaasta ja projektin ohjausryhmästä riippuu, millaisia joustoja aikatauluun, kustannuksiin tai vaatimuksiin voidaan saada neuvoteltua. Projektin johdolta vaaditaan kykyä perustella hyvin joustojen tarpeellisuus ja erityisesti tuoda esiin eri hyötynäkökohdat. Asiakkaan ja ohjausryhmän luottamus on edellytyksenä sille, että projekti voi olla aidosti joustava. Asiakkaan tiivis osallistuminen projektiin on edellytys hankkeen onnistumiselle. Erityisen tärkeää on asiakkaan osallistuminen suunnittelu- ja validointiprosessiin. Asiakkaan edustajan kuuluminen projektin ohjausryhmään on myös suotavaa, sillä se auttaa ohjausryhmää saamaan ajan tasalla olevaa tietoa projektin etenemisestä. Tämä on tärkeää varsinkin silloin, jos ohjausryhmän hyväksyntä tarvitaan vain projektin alussa ja lopussa.

Myös sprintin työlistaan perustuva raportointi tuo projektille tarvittavaa läpinäkyvyyttä. Toteutusvaiheessa työlistaa päivitetään jatkuvasti suoritettujen tehtävien osalta. Sen ohella pidetään ajan tasalla arviota jäljellä olevista tehtävistä ja niiden suorittamiseen tarvittavasta ajasta. Yhteenveto arvioista esitetään graafisesti (Burndown Chart) ja sitä pidetään päivitetyn työlistan ohella kaikkien sidosryhmien nähtävillä.

9.4 Yhdistelmämallin soveltuvuus erityyppisiin projekteihin

Joustavuutta hakevien projektien ohella yhdistelmämalli sopii hyvin käytettäväksi myös sellaisissa projekteissa, joissa aikataulun, kustannusten tai vaatimusten suhteen ei voida joustaa ja joissa halutaan noudattaa puhtaasti vesiputousmallia. Scrumin työkaluja ja menetelmiä voi silti olla mahdollista hyödyntää osittain. Rakennusvaiheessa voi olla mahdollista joustaa eri sprinttien välillä esimerkiksi siten, että jokin ominaisuus, tai osia siitä, voidaan siirtää toteutettavaksi toiseen sprinttiin. Joissakin projekteissa voi olla tarpeen tehdä vain pieniä välijulkaisuja ja suurempi ohjelmaversion julkaisu vasta koko projektin päätteeksi, kun kaikki vaaditut toiminnallisuudet on saatu valmiiksi ja sovitettua yhteen. Projektin tyypistä sekä joustavuuden - tai joustamattomuuden - määrästä riippumatta Scrum-tiimin tiivis yhteistyö sprinttien aikana lisää tehokkuutta ja luovuutta. Päivittäinen Scrum-palaveri parantaa myös tiedonkulkua. Lisäksi suhde asiakkaaseen paranee läheisen yhteistyön ja nopeassa tahdissa julkaistavien ohjelmapakettien myötä. Kuten projektityön tavoitteeksi oli asetettu, tämä projektinhallintamenetelmä noudattaa IT-organisaatiossa toistaiseksi käytettävän PM Methodologyn kehystä, mutta hyödyntää kuitenkin Scrumin parhaita ominaisuuksia, jotka lisäävät prosessin joustavuutta, tehokkuutta sekä kaikkein tärkeimpänä - asiakkaan tyytyväisyyttä.

9.5 Yhdistelmämallin ja PM Methodologyn vertailua

Tärkeimpänä erona yhdistelmämallin ja PM Methodologyn välillä voidaan pitää sitä, että yhdistelmämallia noudattavassa projektissa asiakkaalle saadaan toimitettua uusia julkaisuja lyhyin väliajoin ja ilman keskeytyksiä, kun taas PM Methodologya noudattavassa projektissa asiakas voi joutua odottamaan uutta julkaisua jopa vuoden. Pitkä ajan kuluessa moni asia on voinut jo muuttua, eikä lopputulos mahdollisesti enää vastaakaan asiakkaan tarpeita. Menetelmät eroavat toisistaan selkeästi myös siinä, että muutosten tekeminen yhdistelmämallissa on täysin mahdollista, kunhan pysytellään sovittujen raamien sisällä. Kunkin sprintin työlista suunnitellaan tarkasti vasta sen jäl-

keen, kun tuotteen työlista on ensin priorisoitu. PM Methodologya noudattavassa projektissa muutoksen tekeminen on vaikeaa ja aiheuttaa runsaasti ylimääräistä työtä sekä kustannuksia. Kolmas selkeä ero on siinä, kuinka asiakas osallistuu ja pysyy ajan tasalla projektin edetessä. Yhdistelmämallissa asiakas osallistuu projektiin tiiviisti sen koko keston ajan, mutta PM Methodologyn mukaisessa projektissa asiakas saattaa odotella kuukausia konseptin jäädyttämisen ja testausvaiheen välillä olematta tarkasti selvillä siitä, miten projekti etenee. Lisäksi yhdistelmämallissa projektin kaikki muutkin sidosryhmät pysyvät hyvin ajan tasalla mm. jokaisen sprintin aikana jatkuvasti päivitettävän graafisen yhteenvedon (Burndown Chart) avulla. PM Methodologyn mukaisessa projektissa puolestaan saattaa kulua viikkoja tai jopa kuukausia, ennen kuin kuhunkin merkkipaaluun liittyvät dokumentit ja raportit valmistuvat. Kaiken kaikkiaan voidaan todeta, että mm. tiiviin yhdessä työskentelemisen myötä toiminnan tehokkuus, luovuus ja tiedon kulku ovat yhdistelmämallia noudattavassa projektissa paremmalla tasolla verrattuna PM Methodologya noudattavaan projektiin.

9.6 Tilaajan palaute ja arvio opinnäytetyön tuloksesta

Opinnäytetyön tilaaja totesi palautteessaan, että on syytä painottaa Scrumin mukaisten työlistojen tarkkaa suunnittelua erityisesti silloin, jos projekti on yhteydessä sellaisiin projekteihin, jotka eivät noudata joustavaa menetelmää. Tällöin joustavuudesta voidaan joutua tinkimään mm. aikataulun suhteen. Jokaisen sprintin lopputulos sekä sen varmistaminen testauksilla on myös syytä määritellä tarkasti, vaikka tasapainon löytäminen tarkan määrittelyn ja joustavuuden välillä onkin haasteellista. Lisäksi suuressa projektissa on johtoryhmän tueksi tarpeen muodostaa kunkin sidosryhmän edustajista tekninen asiantuntijaryhmä, johon johtoryhmä voi päätöksiä tehdessään luottaa, sillä suuren projektin johtoryhmän jäsenillä ei välttämättä ole tarkkaa tietämystä vaadituista ominaisuuksista, mutta he ovat kuitenkin vastuussa budjetista.

Tilaaja piti työn lopputulosta hyvänä. Hänen arvioi tutkimuksen olevan hyvin kattava, tulosten analysoimisen perusteellista ja esittämisen selkeää. Hänen mielestään Scrum sopii sellaisenaan vain pieniin tai keskikokoisiin projekteihin, mutta suurissa projekteissa sitä voidaan hyödyntää joillakin yksittäisillä alueilla. Tehty ehdotus oli hänen mielestään ehdottomasti käyttökelpoinen.

10 YHTEENVETO

Kuten jo aiemmin todettiin, ei ole olemassa yksiselitteistä vastausta siihen, millainen projektinhallintamenetelmä olisi paras kullekin projektille. Vaikka projektinhallintamenetelmän avulla voidaan toteuttaa erilaisia projekteja samalla tavoin, toisaalta myös erilaisia menetelmiä tarvitaan erilaisten projektien hallintaan. Menetelmästä riippumatta projektin tärkein päämäärä on kuitenkin aina asiakkaan tarpeiden tyydyttäminen. Jatkuvasti muuttuva ympäristö ja muuttuvat vaatimukset ovat vaikuttaneet siihen, että moni organisaatio on alkanut luopua perinteisistä toimintatavoistaan. Asiakkaan tarpeet on kyettävä tyydyttämään entistä nopeammin. Muuttuviin tilanteisiin on pystyttävä reagoimaan nopeasti ilman, että toiminnan tehokkuus kärsii. Erityisesti ohjelmistokehityksessä muutospaineet ovat suuret ja jotkut organisaatiot ovatkin päätyneet siirtymään projekteissaan kokonaan ketteriin projektinhallintamenetelmiin. Osa on yhdistellyt jotakin vanhaa ja jotakin uutta, eli valikoinut parhaina pitämänsä ominaisuudet suunnitelmapainotteisesta sekä ketterästä ohjelmistokehityksestä ja rakentanut niistä omiin projekteihinsa sopivan hybridimenetelmän.

Opinnäytetyötä varten tehtyjen haastattelujen perusteella voidaan todeta, että suunnitelmapainotteisen ja ketterän projektihallintamenetelmän, kuten PM Methodolgyn ja Scrumin, voi yhdistää toimivaksi hybridimenetelmäksi. Kussakin projektissa on syytä aina harkita tarkasti, millaisessa suhteessa perinteistä ja ketterää menetelmää on järkevää käyttää. Scrumin käyttäminen yhtenä tai useampana komponenttina systeemissä tarjoaa PM Methodologyn käyttöön ketterän ohjelmistokehityksen välineet ja työskentelymenetelmät. Joillekin projekteille tämä voi tosin merkitä vain tarpeellista siirtymävaihetta, kunnes ryhdytään käyttämään yksinomaan ketterää projektinhallintamenetelmää.

Opinnäytetyö valmistui suunnitellussa aikataulussa. Aikaa työn tekemiseen oli riittävästi siitäkin huolimatta, että aikatauluja jouduttiin välillä sovittelemaan mm. työkiireiden vuoksi ja haastattelujen tekeminen jakautui suhteellisen pitkälle aikavälille. Haastattelut vahvistivat hyvin sitä kuvaa, mikä opinnäytetyön tekijälle alkoi hahmottua jo ensimmäisten keskustelujen aikana, jotka hän kävi työn tilaajan kanssa. Vaikka kirjallista materiaalia vesiputousmallin ja Scrumin yhdistämisestä ei löytynytkään, aiheeseen sopivaa kirjallisuutta oli kuitenkin runsaasti saatavilla. PM Methodologyn osalta dokumentaatio jätettiin aiheen laajuuden vuoksi käsittelemättä tarkemmin.

LÄHTEET

Agile Manifesto. 2001. Saatavissa: http://agilemanifesto.org [viitattu 25.04.2009].

Berkun, S. 2006. Projektinhallinnan taito. 1. painos. Jyväskylä: Gummerus Kirjapaino Oy.

Boehm, B. & Turner, R. 2008. Balancing Agility and Discipline: A guide for the Perplexed. 6. painos. Boston: Addison-Wesley.

Haikala, I. & Märijärvi, J. 2004. Ohjelmistotuotanto. 10. uudistettu painos. Helsinki: Talentum Media Oy.

Kettunen, P. 2009. Agile Software Development In Large Scale New Product Development Organization: Team Level Perspective. Väitöskirja 5.11.2009. Helsinki: Teknillinen korkeakoulu. Saatavissa: http://lib.tkk.fi/Diss/2009/isbn9789522481146/isbn9789522481146.pdf [viitattu 15.12.2009].

Krasselt, R. Ohjelmapäällikkö. Puhelinhaastattelu 10.3.2009. Nokia Siemens Networks.

Krasselt, R. Ohjelmapäällikkö. Sähköposti 10.3.2009. Nokia Siemens Networks.

Kuikka, M. 2008. Ohjelmistoprosessi aloittavassa ohjelmistoyrityksessä. Pro Gradu-tutkielma 24.5.2008. Joensuu: Joensuun yliopisto. Saatavissa: ftp://cs.joensuu.fi/pub/Theses/2008_MSc_Kuikka_Mika.pdf [viitattu 20.03.2009].

Litke, H. & Kunow, I. 2004. Projektinhallinta. 1. painos. Helsinki: Oy Rastor Ab.

Murch, R. 2002. IT-projektinhallinta. 1. painos. Helsinki: Edita Prima Oy.

Pietarila, J. Projektipäällikkö. Sähköposti 6.4.2009. Nokia Siemens Networks.

Pohjonen, R. 2002. Tietojärjestelmien kehittäminen. 1. painos. Jyväskylä: Docendo Finland Oy.

Project Management Methodology. Nokia Siemens Networks. Saatavissa: https://inside.nokiasiemensnetworks.com/global/work/processes_and_tools/process_it manage-
ment/project_management_methodology_for_nsn_be_and_it/pages/overview.aspx [viitattu 18.5.2009].

Rissanen, T. 2002. Projektilla tulokseen. 1. painos. Jyväskylä: Gummerus Kirjapaino Oy.

SCPR – Solution Cost and Planning Reporting. Nokia Siemens Networks. Saatavissa: https://scpr.inside.nokiasiemensnetworks.com/SCPR [viitattu 17.8.2009].

Scrum, Eclipse Process Framework Wiki. Eclipse. Saatavissa: http://epf.eclipse.org/wikis/scrum/Scrum/customcategories/scrum_overview_A34DD E47.html [viitattu 15.4.2009].

Scrum, Ketterät käytännöt.fi. Sininen Meteoriitti. Saatavissa: http://www.ketteratkaytannot.fi/fi-FI/Menetelmat/Scrum [viitattu 25.4.2009].

Tusa, M. Ohjelmapäällikkö. Puhelinhaastattelut 26.03.2009, 6.4.2009 ja 23.6.2009. Nokia Siemens Networks.

Vahter, P. Projektipäällikkö. Sähköpostit 18.3.2009, 20.3.2009, 1.4.2009 ja 20.4.2009. Nokia Siemens Networks.

What is Scrum?, ScrumAlliance. Saatavissa: http://www.scrumalliance.org/pages/what_is_scrum [viitattu 3.5.2009].