Päivi Juka-Koski

PROJEKTINHALLINNAN KEHITTÄMINEN SCRUMILLA

Käyttöönottoselvitys pienessä IT-alan yrityksessä

PROJEKTINHALLINNAN KEHITTÄMINEN SCRUMILLA

Käyttöönottoselvitys pienessä IT-alan yrityksessä

Päivi Juka-Koski
Opinnäytetyö
Kevät 2011
Tietojenkäsittelyn koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma

Tekijä: Päivi Juka-Koski
Opinnäytetyön nimi: Projektinhallinnan kehittäminen Scrumilla – Käyttöönottoselvitys pienessä IT-alan yrityksessä
Työn ohjaaja: Pekka Ojala
Työn valmistumislukukausi ja -vuosi: Kevät 2011 Sivumäärä: 93 + 8

Opinnäytetyön kehittämistehtävänä oli perehtyä ketteriin ohjelmistokehitysmenetelmiin kuuluvaan Scrumiin ja sen käyttöönottoon. Hyvän kokonaisnäkemyksen muodostamiseksi tutustuttiin lyhyesti myös muihin ohjelmistokehitysmenetelmiin. Saadun tietämyksen pohjalta pohdittiin Scrumin käyttöönottoa toimeksiantajayrityksessä. Opinnäytetyön toimeksiantajana oli nimettömänä pysyttelevä pieni IT-alan yritys, jonka ohjelmistoprojektien toteutukseen kaivattiin uusia näkökulmia.

Opinnäytetyön viitekehyksessä käsitellään lyhyesti perinteisiä ja ketteriä ohjelmistokehitysmenetelmiä ja laajemmin Scrumia ja sen käyttöönottoa. Käytetty lähdeaineisto koostuu pääasiassa englanninkielisistä kirjoista sekä digitaalisista artikkeleista ja julkaisuista.

Kehittämistehtävän perustana käytettiin monipuolisen teoria-aineiston lisäksi toimeksiantajan edustajille tehtyjä haastatteluja sekä opinnäytetyön tekijän tekemiä havaintoja hänen työskennellessään kyseisessä yrityksessä. Näin saatiin muodostettua näkemys yrityksen projektinhallinnan lähtökohtatilanteesta, havaituista haasteista ja asetetuista odotuksista.

Kehittämistehtävän tuloksena todetaan, että Scrum ei kaikilta osin sovellu toimeksiantajayritykselle. Tämä johtuu esimerkiksi yrityksen projektitiimien pienuudesta ja käyttöönottoon käytettävissä olevien henkilöstöresurssien vähyydestä. Scrumista löytyy kuitenkin joitakin käytäntöjä, joiden avulla haasteisiin ja odotuksiin voidaan vastata sekä yrityksen ohjelmistokehityskäytäntöjä ja projektitiimien toimintaa tehostaa. Nämä tulokset eivät ole yleistettävissä muihin yrityksiin, koska jokaisen organisaation toiminta on erilaista ja erilaisia lähestymistapoja vaativaa.

Asiasanat: Scrum, ohjelmistokehitysmenetelmä, ketterät menetelmät, ohjelmistokehitys, projektinhallinta, käyttöönotto

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Business Information Systems

Author: Päivi Juka-Koski
Title of thesis: Developing project management with Scrum – Feasibility study of adoption of Scrum in a small IT company
Supervisor: Pekka Ojala
Term and year when the thesis was submitted: Spring 2011 Number of pages: 93 + 8

The aim of the thesis was to get acquainted to an agile software development method called Scrum and to its adoption. To gain a good overall view also other software development methods were familiarized. After gaining a good knowledge of the subject, the task was to consider the adoption of Scrum in a small IT company who was also the client of this thesis. The company wanted to stay anonymous.

The context briefly discusses the traditional and agile software development methods. After the short introduction to the methods, the focus moves on Scrum and its adoption. The source material used consists mainly of English-language books, articles, and publications found on the Internet.

Along with diverse theory, also interviews of the people working in the client company and the author's own observations of the company were used. This way an understanding of company's project management methods, noticed challenges and expectations was formed.

The result of the development task is that Scrum cannot be used in the company as such. Some parts of it could be used but in general, the adoption of Scrum is not possible. This is because of, among other things, the small size of the company and the small amount of human resources available for the adoption. There are however some features in Scrum which can respond to the challenges and expectations of the company and improve its software development methods and the effectiveness of its project teams. These results cannot be generalized to other companies because each organization works differently and requires different approach of how to develop their software development methods.

Keywords: Scrum, software development method, agile methods, software development, project development, adopting Scrum

SISÄLLYS

1 JOHDANTO

Ohjelmistokehitysmenetelmät määrittelevät, missä järjestyksessä ja miten projektin vaiheet etenevät (Boehm 1988, hakupäivä 3.3.2011). Nämä menetelmät voidaan jakaa perinteisiin ja ketteriin menetelmiin. Perinteiset menetelmät pohjautuvat yksityiskohtaisiin suunnitelmiin ja perusteelliseen dokumentaatioon (Deemer, Benefield, Larman & Vodde 2010, 3). Ketterät menetelmät ovat perinteisiä menetelmiä kevyempiä. Ne tarjoavat ohjelmistokehityksen ratkaisuja erityisesti Internetissä tai mobiililaitteissa käytettäviä ohjelmistoja kehittäville yrityksille, joiden ohjelmistokehitysprosessin tulee olla kevyttä, nopeaa ja valpasta. (Abrahamsson, Salo, Ronkainen & Warsta 2002, 3.)

Ketteristä menetelmistä suosituin on Scrum (Deemer ym. 2010, 4). Se keskittyy parantamaan Scrum-tiimiksi kutsutun tuotekehitystiimin toimintaa, jolloin ohjelmistoja voidaan tuottaa joustavasti jatkuvasti muuttuvassa toimintaympäristössä (Abrahamsson ym. 2002, 27). Scrum ja sen käyttöönotto perustuvat empirismiin eli kokemusperäiseen tietoon (Schwaber & Beedle 2001, 25). Tämän vuoksi käyttöönottoprosessi paljastaa eri organisaatioissa erilaisia ohjelmistokehitykseen liittyviä heikkouksia ja kehittämisen kohteita, joihin sekä yrityksen johto että työntekijät voivat omalla toiminnallaan vaikuttaa (Deemer ym. 2010, 19).

Tämän opinnäytetyön tavoitteena on tarjota katsaus perinteisiin ja ketteriin ohjelmistokehitysmenetelmiin, painottuen erityisesti Scrumiin ja sen käyttöönottoon. Työn kehittämistehtävänä on pohtia Scrumin soveltuvuutta toimeksiantajan ohjelmistokehitysmenetelmäksi ja antaa yrityksen johdolle uusia näkökulmia ohjelmistokehitysprojektien hallinnan kehittämiseksi.

Opinnäytetyön toimeksiantajana on nimettömänä pysyttelevä, Pohjois-Suomessa toimiva IT-alan yritys. Se on perustettu 2000-luvulla ja se työllistää yrittäjien lisäksi kolmesta viiteen työntekijää. Yrityksen toiminta jakaantuu asiakkaiden tietojärjestelmien ja laitteistojen ylläpito- ja tukityöhön sekä ohjelmistokehitystyöhön. Sekä ylläpito- että ohjelmistokehitystehtävissä on omat työntekijänsä, joilla on koulutusta ja ammattitaitoa juuri kyseiselle alalle. Ohjelmistopuolella toteutetaan pääasiassa verkkopohjaisia ohjelmistoja, jotka räätälöidään asiakkaiden tarpeisiin. Tässä käytetään projektiluontoista lähestymistapaa, mutta yrityksessä ei ole käytössään mitään tiettyä ohjelmistokehitysprojektien hallintamenetelmää. (Yrittäjä 4.12.2009, haastattelu; Projektipäällikkö 15.4.2011, haastattelu).

Opinnäytetyön aiheen valintaan vaikuttivat sekä toimeksiantajayrityksessä nähty tarve projektinhallinnan kehittämiseen että työn tekijän oma kiinnostus aihepiiriin. Projektipäällikön mukaan yrityksen johdolla ei ole koulutusta projektien hallinnasta. Tarpeesta huolimatta johtajilla ei ole ollut resursseja perehtyä ohjelmistokehitysprojektien hallintaan ja systemaattiseen kehittämiseen. (Projektipäällikkö 15.4.2011, haastattelu). Opinnäytetyön aiheen voidaankin katsoa olevan yritykselle tärkeä, ja opinnäytetyöraportin avulla toimeksiantajayrityksen johto voi perehtyä eri ohjelmistokehitysmenetelmiin ja siihen, miten yrityksen projektinhallintaa voidaan parantaa.

Opinnäytetyön tekijä on suuntautunut opinnoissaan ohjelmistosuunnitteluun, ja lisäksi hän on opiskellut esimiestyöhön ja johtamiseen liittyviä opintojaksoja. Aihevalinta täydentää näitä opintoja ja tuo tekijälle arvokasta lisätietoa ohjelmistoprojektien hallinnasta. Scrum on herättänyt yleisesti kiinnostusta IT-alan yrityksissä ja sitä hyödynnetään useiden yritysten projektinhallinnassa. Tästä johtuen työn tekijä kokee aiheeseen perehtymisen antavan hänelle lisävalmiuksia, joista voi olla hyötyä tulevan työuran aikana.

Teoriataustassa on hyödynnetty monipuolisesti painettuja kirjalähteitä, artikkeleita ja luotettaviksi katsottuja digitaalisia lähteitä sekä ScrumMaster-koulutuksessa suullisesti kerrottuja tietoja. Varsinaisen toimeksiantajalle toteutetun käyttöönottosuunnitelman pohjana ovat toimeksiantajayrityksen edustajille tehdyt haastattelut sekä opinnäytetyön tekijän omat havainnot hänen työskennellessään aikaisemmin toimeksiantajayrityksessä.

2 OHJELMISTOKEHITYSMENETELMÄT

Abrahamsson, Salo, Ronkainen ja Warsta viittaavat julkaisussaan Hawryshin ja Ruprechtin kantaan siitä, että sama ohjelmistokehityksen projektinhallintamenetelmä ei voi toimia kaikissa erilaisissa ohjelmistoprojekteissa. Tämän vuoksi projektipäällikön tulee tunnistaa kunkin projektin vaatimukset ja valita sen mukaisesti parhaaksi katsomansa käytettävissä oleva kehitysmenetelmä. Julkaisussa viitataan myös McCauleyhin, jonka mukaan käytännön työssä on tarvetta sekä ketterille että prosessiohjautuneille menetelmille. (2002, 12–13.) Nämä näkökulmat huomioon ottaen organisaatioiden johdon ja projektipäälliköiden tulisi tutustua eri menetelmiin, suhtautua niiden käyttöön avoimin mielin ja tarvittaessa soveltaa niitä käytännön projektityössä.

Ideaalimaailmassa kehitystyötä tehdään selvittämällä ensin asiakkaan vaatimukset, minkä jälkeen vaatimukset analysoidaan, ja niiden pohjalta tehdään suunnitelma, joka lopuksi toteutetaan ja testataan. Tämän jälkeen tuotteen pitäisi olla valmis luovutettavaksi asiakkaalle. (Schach 2007, 35–36.) Käytännössä ohjelmistokehitys ei kuitenkaan etene näin suoraviivaisesti. Schach nostaa esiin kaksi syytä. Ensinnäkin ohjelmistoalan ammattilaiset tekevät virheitä – kuten kaikki muutkin asiantuntijat. Toiseksi asiakkaan vaatimukset voivat muuttua kehitystyön edetessä, mikä vaikuttaa vääjäämättä suunnitelmiin ja toteutukseen. (2007, 36.)

Ohjelmistokehitysmallit tarjoavat ratkaisuja siihen, missä järjestyksessä ja miten projektin pääasialliset vaiheet tulee ratkaista (Boehm 1988, hakupäivä 3.3.2011). Tämä ei ole kustannustenkaan kannalta yhdentekevää. Projektin alkupuolen vaatimus-, määrittely- tai suunnitteluvaiheissa havaitut muutokset ovat vielä suhteellisen edullisia korjata. Kustannukset kasvavat kuitenkin suuriksi, jos muutoksia tehdään sen jälkeen kun ohjelmistoa on ohjelmoitu tai, pahimmassa tapauksessa, kun ohjelmisto on jo toimitettu asiakkaalle. (Schach 2007, 50–51.)

Erilaiset ohjelmistokehitysprojektien hallintamenetelmät voidaan jakaa perinteisiin menetelmiin ja ketteriin menetelmiin. Perinteisiä menetelmiä ovat esimerkiksi vesiputousmalli, protoilumalli, spiraalimalli, V-malli ja evo-malli (Haikala & Märijärvi 2006, 42–45, 288). Ketteriä menetelmiä ovat kolmannessa luvussa tarkemmin käsiteltävän Scrumin lisäksi muun muassa Extreme Programming (XP), Crystal Methods, Feature-Driven Development, Rational Unified Process, Dynamic Systems Development Method (DSDM), Adaptive Software Development ja Open Source Software Development (Abrahamsson ym. 2002, 18).

2.1 Perinteiset menetelmät

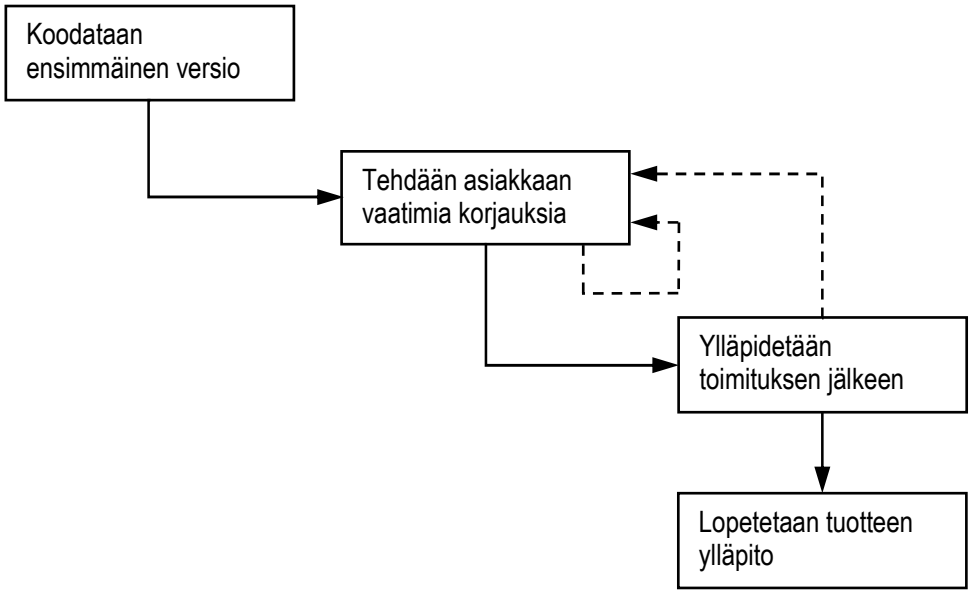
Ohjelmiston elinkaareksi (life cycle) kutsutaan aikaa, joka kuluu ohjelmiston kehittämisen aloittamisesta sen käytöstä poistamiseen (Haikala & Märijärvi 2006, 36). Perinteisiin ohjelmistokehitysmenetelmiin liittyvä termi elinkaarimalli (Life Cycle Model) tarkoittaa ohjelmistotuotteen kehittämisen ja ylläpidon aikana toistettavia vaiheita (Schach 2007, 38).

Tyypillinen perinteisten menetelmien mukainen projekti alkaa yksityiskohtaisten suunnitelmien laatimisella. Nämä suunnitelmat dokumentoidaan perusteellisesti. Tämän jälkeen määritellään välttämättömät tehtävät työn toteuttamiseksi ja arvioidaan niiden vaatima aika. Kun tarvittavat sidosryhmät ovat hyväksyneet suunnitelman, tuotekehitystiimi aloittaa työskentelynsä. Tiimin jäsenet toteuttavat vuorollaan oman erikoistumisalansa mukaiset toimet tuotteen osa-alueen valmistumiseksi. Lopuksi tuote testataan ja toimitetaan asiakkaalle. Prosessin aikana seurataan tarkasti, että tuotetta tehdään suunnitelmista poikkeamatta. Tällä pyritään varmistamaan se, että lopputuote on asiakkaan hyväksymien suunnitelmien mukainen. (Deemer, Benefield, Larman & Vodde 2010, 3.)

Perinteisten menetelmien vahvuutena on se, että ne ovat äärimmäisen loogisia: kehitystyössä ajatellaan ennen kuin toteutetaan, kirjoitetaan kaikki tarvittavat tiedot ylös, noudatetaan suunnitelmia ja pidetään kaikki mahdollisimman organisoituna. Heikkoutena nähdään esimerkiksi se, että usein suunnitteluvaiheessa ei ole tiedossa kaikkia projektin edetessä ilmeneviä asioita, eikä niitä näin ollen voida ottaa huomioon. (Deemer ym. 2010, 3.)

2.1.1 Koodaa ja korjaa -malli

Yksi ohjelmistokehityksen varhaisimmista malleista, koodaa ja korjaa (Code-and-Fix Model), sisältää kaksi vaihetta: koodin kirjoittamisen ja koodin ongelmien korjaamisen. Tämän mallin mukaan ensin kirjoitetaan koodia, ja vasta sen jälkeen mietitään ohjelmiston vaatimuksia, suunnittelua, testausta ja ylläpitoa. (Boehm 1988, hakupäivä 3.3.2011.) Malli on kuvattu kuviossa 1. Tämä äkkiseltään helpolta kuulostava tapa on edelleen varsin houkutteleva ohjelmoinnista innostuneille kehittäjille, jotka haluavat toteuttaa ohjelmistossa omia visioitaan. Tähän voidaan päätyä erityisesti silloin, kun kehitystiimin tehokkuutta arvioidaan koodirivien määrällä (Schach 2007, 51). Tällä tavalla eletään hetkellisesti harhaluulossa, että on saatu paljon aikaan, mutta projektin edetessä koodaa ja korjaa -mallin mukainen toimintatapa kostautuu.



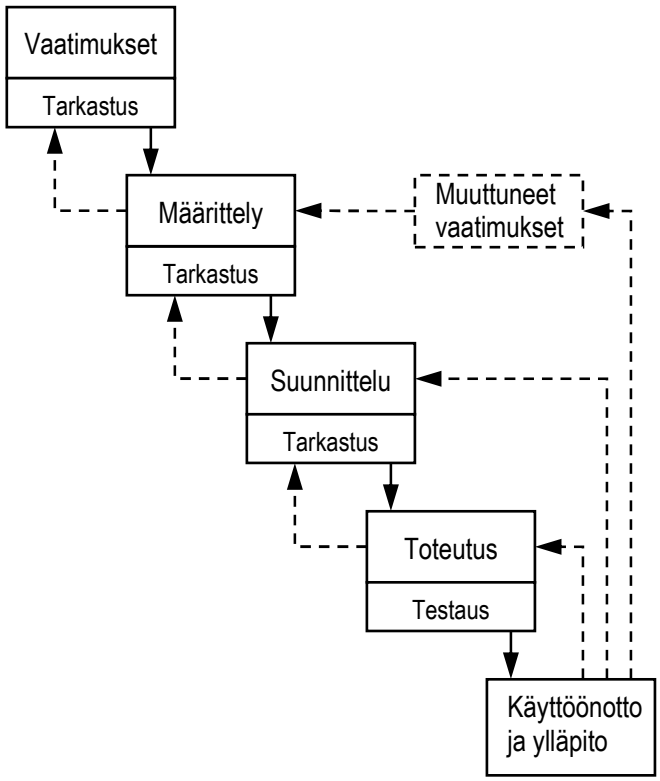
KUVIO 1. Koodaa ja korjaa -malli (Schach 2007, 50, käännetty vapaamuotoisesti suomeksi)

Schach huomauttaa, että koska korjaa ja koodaa -mallissa ohjelmistokehitystä lähdetään viemään eteenpäin ilman riittävää ennakkosuunnittelua, koodia joudutaan muokkaamaan useita kertoja ennen kuin saadaan aikaiseksi asiakasta tyydyttävä lopputulos. Tämä vaatii enemmän työtunteja ja tulee siten huomattavasti kalliimmaksi kuin huolellinen vaatimusten selvittäminen ja suunnittelu. (2007, 50–51.) Boehm nostaa kustannusten ohella esille myös sen, että useiden korjausten myötä koodin rakenne voi heikentyä. Lisäksi asiakkaan tarpeet eivät tule riittävästi huomioiduiksi, joten vaarana on, ettei lopputulos vastaa asiakkaan odotuksia. (1988, hakupäivä 3.3.2011.) Schach kertoo myös, että koska määrittely- ja suunnitteludokumentteja ei ole tehty tai pidetty ajan tasalla, tuotteen ylläpito voi olla erittäin vaikeaa. Näiden seikkojen vuoksi koodaa ja korjaa -mallia ei pidetä hyvänä ohjelmistokehitysmenetelmänä, vaan yrityksessä tulisi aina toimia jonkin muun menetelmän mukaisesti. (2007, 51.)

2.1.2 Vesiputousmalli

Klassisin elinkaarimalleista on niin sanottu vesiputousmalli (Waterfall Model), jonka tiettävästi ensimmäisen kerran esitti Winston W. Royce vuonna 1970 (Haikala & Märijärvi 2006, 36). Schach (2007, 392) kuvaa vesiputousmallia lineaarisena, vaiheesta toiseen siirtyvänä mallina, jossa voidaan tarvittaessa edetä vaihe kerrallaan taaksepäin, mikäli jossain myöhemmässä vaiheessa havaitaan ongelmia. Vesiputousmallista on olemassa useita eri muunnelmia, joissa yleensä on ainakin määrittely-, suunnittelu- ja toteutusvaiheet (Haikala & Märijärvi 2006, 36).

Yksi vesiputousmallin muunnelmista on kuvattu kuviossa 2. Aluksi asiakkaalta saatujen vaatimusten pohjalta vastataan kysymyksiin mikä on ratkaistava ongelma, onko ratkaisua olemassa, mitä se saa maksaa, mitä reunaehtoja sillä on ja niin edelleen. Määrittelyvaiheessa pohditaan sitä, millainen ohjelmisto täyttää vaatimukset. Seuraavaksi suunnitellaan, miten vaatimusten mukainen ohjelmistotuote toteutetaan. Toteutusvaiheessa tehdään varsinainen ohjelmointityö. Laadunvarmistustoimenpiteitä, kuten testaamista, katselmointia ja tarkastuksia, tehdään jokaisen vaiheen aikana ja jälkeen. Tällä pyritään puuttumaan virheisiin mahdollisimman aikaisessa vaiheessa. Lopuksi toteutetaan käyttöönotto, jonka jälkeen ohjelmistoa ylläpidetään ja siitä tehdään tarvittaessa uusia versioita muuttuneiden vaatimusten pohjalta. (Haikala & Märijärvi 2006, 36–37.)



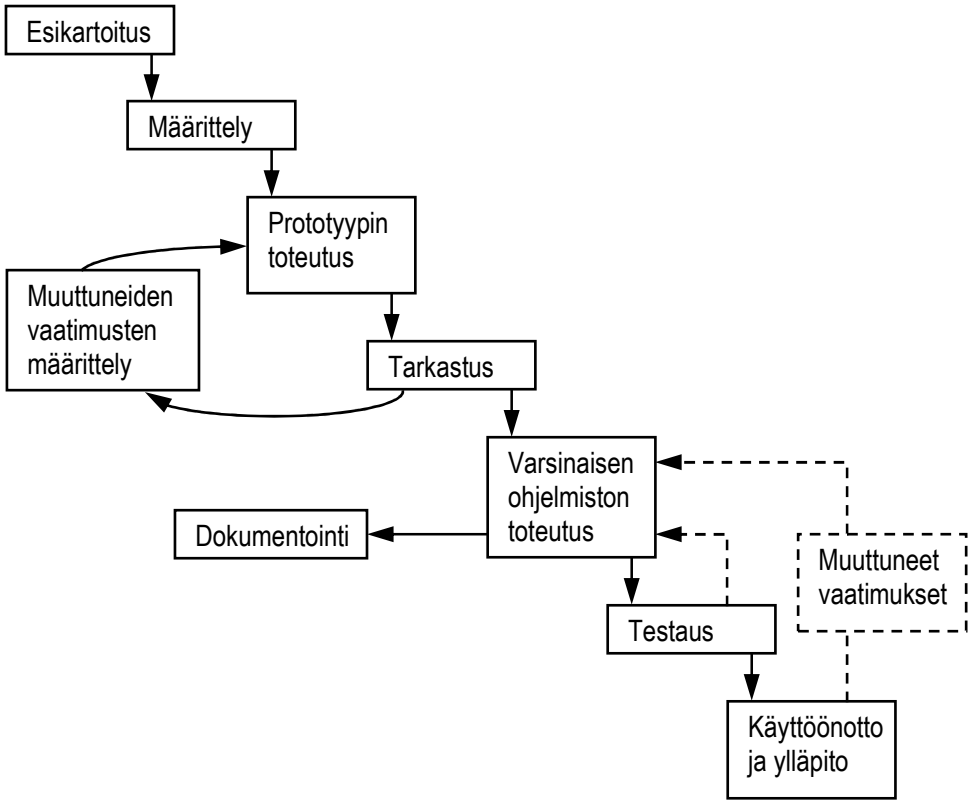
KUVIO 2. Esimerkki vesiputousmallista (mukailtu lähteistä Haikala & Märijärvi 2006, 36 ja Schach 2007, 51)

Vesiputousmallin vahvuutena on sen kurinalaisuus: seuraavaan vaiheeseen ei edetä ennen kuin edeltävän vaiheen dokumentaatio on täysin valmis ja siihen mennessä tehty ohjelmisto on testattu ja hyväksytty (Schach 2007, 51–52). Ongelmana voidaan nähdä mallin perusajatus siitä, että jo määrittely- ja suunnitteluvaiheessa tehdään sitovia päätöksiä, joiden pohjalta kehittämistyötä toteutetaan käytännössä. Toteuttamisvaiheen edetessä havaitaan lähes poikkeuksetta

asioita, joita ei suunniteltaessa ole osattu ottaa huomioon. Tämän seurauksena voi olla suuri määrä käyttökelvotonta koodia, jota ei voida hyödyntää uusien suunnitelmien ja määritelmien jälkeen. (Boehm 1988, hakupäivä 3.3.2011.) Myös Haikala ja Märijärvi toteavat, että käytännössä ohjelmistokehitys ei voi edetä täysin vesiputousmallin mukaisesti, koska osa vaatimuksista selviää vasta projektin aikana ja vaatimukset voivat myös muuttua. Heidän mukaansa ohjelmisto-kehitysprojekteissa voidaan kuitenkin pyrkiä tämän mallin mukaiseen toimintaan siinä määrin kuin se on mahdollista. (2006, 41.)

2.1.3 Protoilumalli

Protoilu- eli prototyyppimallissa (Prototype Model) noudatetaan osittain vesiputousmallista tuttuja vaiheita, mutta projektin aluksi toteutetaan nopealla syklillä prototyyppi eli jokin kehitettävän ohjelmiston avaintoiminnallisuuksista. Asiakas ja ohjelmiston tulevat käyttäjät voivat kokeilla prototyyppiä ja olla siten vuorovaikutuksessa toteutetun ohjelmiston osan kanssa. Näin asiakas voi antaa palautetta toiminnallisuuksista ja kehittäjät voivat jatkaa varsinaista kehitystyötä tietäen, että tuote vastaa asiakkaan todellisia tarpeita. Ohjelmistokehitystä jatketaan etenemällä vaati-muksista niiden analysointiin, toteutettavan ohjelmiston suunnitteluun ja toteutukseen, ja lopulta ohjelmiston toimitukseen ja ylläpitoon. (Schach 2007, 53.) Yksi esimerkki protoilun etenemisestä on kuvattu kuviossa 3.



KUVIO 3. Esimerkki protoilumallista (mukailtu lähteistä Haikala & Märijärvi 2006, 46 ja Schach 2007, 53)

Protoilumalli etenee pääasiassa lineaarisesti kuten vesiputousmallikin. Verrattaessa näitä kahta mallia, protoilumallin etuna on, että projektissa ei tarvitse yhtä todennäköisesti palata takaisin täsmentämään aikaisempia vaiheita. Ohjelmistokehittäjät ovat tehneet toteutettavan tuotteen määritelmät prototyypistä saadun palautteen avulla. Näin ollen todennäköisyys siihen, ettei vaatimuksia tarvitse muuttaa projektin edetessä, kasvaa. Suunnitelmia on myös saatu jo työstettyä ja kehitystiimillä on selvillä vähintäänkin se, mitä tuotteelta ei haluta. Vaikka prototyyppi kuvaakin vain tiettyä ohjelmiston osaa, antaa sen parissa tehty työ kehittäjille käsityksen lopullisen ohjelmiston toteuttamisesta. (Schach 2007, 54.)

Oleellista protoilussa on se, että prototyyppi toteutetaan mahdollisimman nopealla aikataululla, jotta varsinainen kehitystyö pääsee tehokkaasti käyntiin. Prototyypin toteuttajien tulee muistaa, että tavoitteena on saada selville asiakkaan todelliset tarpeet ilman että prototyypin viimeistelyyn käytetään liikaa aikaa. Prototyypissä tuodaan esille asiakkaalle näkyvät tärkeimmät toiminnallisuudet, kuten lomakkeet ja raportit. Asiakkaalle "piilossa" olevat toiminnallisuudet, kuten virheilmoitukset ja syötteiden tarkistukset, jätetään tässä vaiheessa tekemättä. Esimerkiksi ohjelmiston kaatuminen tai käyttöliittymän asettelujen epätäydellisyys ei haittaa, sillä ei ole tarpeen testata

valmiilta vaikuttavaa tuotetta. (Schach 2007, 53–54, 320–321.) Tällöin vältytään siltä, että asiakas ei luule lähes oikealta näyttävän, mutta oikeasti hyvin keskeneräisen, ohjelmiston olevan käytännöllisesti katsoen valmis. On vain hyvä, että asiakaskin huomaa ohjelmiston keskeneräisyyden ja ymmärtää näin paremmin, että lopullisen tuotteen valmistuminen vaatii vielä lukuisia työtunteja. (Haikala & Märijärvi 2006, 43.)

Projektin alussa voidaan toteuttaa useampiakin prototyyppejä, mikäli se nähdään tarpeelliseksi asiakkaan vaatimusten kartoittamiseksi. Tämän vuoksi prototyyppi kannattaa suunnitella siten, että siihen on mahdollista tehdä helposti muutoksia. (Schach 2007, 321.) Täytyy kuitenkin varoa sitä, että prototyyppiä ei yritetä parannella loputtomasti. Projektissa voidaankin varata prototyyppien kehittämiseen tietty aika, jonka jälkeen edetään varsinaisen ohjelmiston kehittämiseen. (Haikala & Märijärvi 2006, 43–44.)

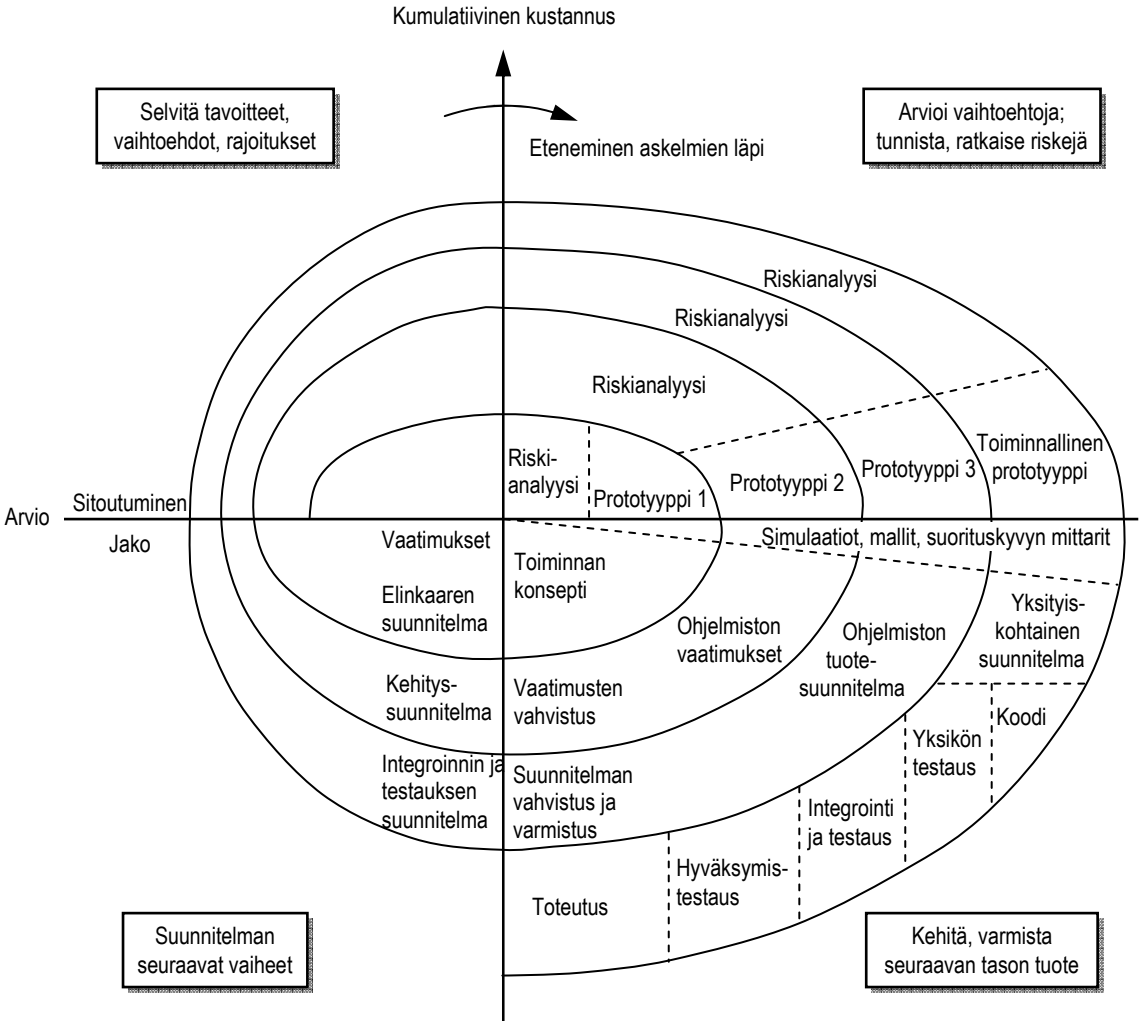
2.1.4 Spiraalimalli

Spiraalimalli (Spiral Model) on Barry W. Boehmin vuonna 1988 esittelemä ohjelmistokehitysmalli. Mallia voidaan hyödyntää ohjelmiston eri osioiden samanaikaisesti tapahtuvaan kehitystyöhön erillisiä spiraaleja seuraamalla tai mallin avulla voidaan hallita ohjelmistoprojektia yhtenä kokonaisuutena. (Boehm 1988, hakupäivä 3.3.2011.)

Spiraalimalli hyödyntää sitä edeltävien ohjelmistokehitysmenetelmien parhaita puolia. Verrattuna aikaisemmin luotuihin, dokumentointia tai ohjelmointia korostaviin ohjelmistokehitysmenetelmiin, spiraalimalli korostaa riskien hallintaa. Siinä pyritään virheiden ja ei houkuttelevien vaihtoehtojen karsimiseen jo varhaisessa vaiheessa. (Boehm 1988, hakupäivä 3.3.2011.) Lisäksi siinä huomioidaan aikainen vaatimusten ymmärtäminen ja niiden toteutettavuuden mallintaminen prototyyppien avulla (Forsberg, Mooz & Cotterman 2000, 24).

Kuviossa 4 on esitetty spiraalimalli. Malli saattaa ensisilmäyksellä olla hämmentävä, koska se ei etene kuluneen ajan mukaisesti perinteisesti vasemmalta oikealle (Forsberg, Mooz & Cotterman 2000, 24). Projektin vaiheiden eteneminen lähtee liikkeelle spiraalin keskiöstä, ja jokaisella neljällä kierroksella toistetaan neljän lohkon sisällä kuvatut vaiheet. Mallin pystysuuntainen akseli kuvaa vaiheiden mukaisesti kasvavaa kumulatiivista kustannusta. (Boehm 1988, hakupäivä 3.3.2011.)

.



KUVIO 4. Spiraalimalli (Boehm 1988, hakupäivä 3.3.2011, *termien suomentamisessa on hyödynnetty lähdettä Forsberg, Mooz & Cotterman 2000, 207)*

Jokainen spiraalin kierroksista alkaa yllä olevan kuvion vasemmasta alakulmasta kyseisen kierroksen tavoitteiden tunnistamisella. Tämän jälkeen arvioidaan tavoitteisiin ja rajoitteisiin liittyvät vaihtoehdot. Mikäli tässä vaiheessa huomataan sellaisia epävarmuustekijöitä, joista voi aiheutua projektille merkittäviä riskejä, tulee muodostaa kustannustehokas strategia riskien toteutumisen välttämiseksi. Tämä voidaan toteuttaa esimerkiksi prototyypeillä, simuloinneilla, benchmarkingilla, analysoinnilla tai näiden ja muiden riskienhallintamenetelmien yhdistelmällä. Kun riskit on tunnistettu ja ratkaistu, toteutetaan kyseisen kierroksen kehittämistehtävät ja vahvistetaan tehdyt vaiheet katselmoinnilla, johon osallistuvat lopulliseen tuotteeseen liittyvät avainhenkilöt. Tällä varmistetaan se, että kaikki asianosaiset ovat sitoutuneet projektiin ennen seuraavaan vaiheeseen siirtymistä. Katselmoinnissa voidaan käydä pienimuotoisesti läpi esimerkiksi yksittäisen ohjelmoijan toteuttama ohjelmiston komponentti, mutta suurimittaista katselmointia ei tarvita. Neljännen kierroksen lopuksi ohjelmistotuote ohjelmoidaan, testataan ja toteutetaan yksityis-

kohtaisten suunnitelmien pohjalta. Ohjelmiston jatkokehitystä ja ylläpitoa toteutetaan uuden spiraalin avulla esiin nousseiden tarpeiden mukaisesti. (Boehm 1988, hakupäivä 3.3.2011.)

Forsbergin ym. mukaan spiraalimallin esitystapa hämärtää kehittämisen hallitsemiseen tarvittavia tarkastuksia. Toisena huonona puolena he näkevät sen, että riskienhallinta-analyysit viivyttävät vähäriskisen tuotteen kehittämistä. Mallissa ei hyödynnetä vaihtoehtoa, jossa riskienhallintaa toteutetaan jatkuvasti kehittämisprojektin rinnalla. (2000, 24.) Spiraalimallin kehittäjä Boehm nostaa esille sen, että malli toimii hyvin sisäisissä ohjelmistokehitysprojekteissa, mutta vaatii jatkokehitystä toimiakseen sopimuksiin perustuvissa projekteissa. Lisäksi malli nojaa vahvasti ohjelmistokehittäjien kykyyn tunnistaa, hallita ja dokumentoida projektiin liittyviä riskejä. Mikäli kehittäjät eivät ole kokeneita, riskien arviointi voi olla hankalaa. (1988, hakupäivä 3.3.2011.)

2.2 Ketterät menetelmät

Ketteryydellä tarkoitetaan valmiutta ja kykyä nopeaan muutokseen (Abrahamsson ym. 2002, 9). Ketterät ohjelmistokehitysmenetelmät huomioivat tuotekehitykseen vaikuttavat realiteetit – oppimisen, innovoinnin ja muutoksen – joiden uskotaan johtavan parempaan lopputulokseen. Ketterissä menetelmissä pyritään perusteellisen suunnittelun ja dokumentoinnin sijasta toimittamaan toimivia ohjelmiston osia asiakkaalle jo projektin varhaisessa vaiheessa. Lisäksi niissä keskitytään päätösvaltaisen, monipuolista ammattilaisista ja asiakkaan edustajasta koostuvan tiimin toiminnan parantamiseen. (Deemer ym. 2010, 4.) Ketterät menetelmät eivät ole mikään uusi ilmiö projektinhallinnassa. Esimerkiksi kolmannessa luvussa tarkemmin käsiteltävää Scrumia on käytetty jo 1990-luvun puolivälistä lähtien. Nämä menetelmät ovat kuitenkin saavuttaneet entistä suurempaa huomiota 2000-luvulle tultaessa. (Leffingwell & Smits 2005, 7.)

Vuoden 2001 helmikuussa 17 niin sanottujen kevyiden menetelmien (Light Methodologies) kehittäjää ja puolestapuhujaa tapasi toisensa Utahissa Yhdysvalloissa. Heidän tavoitteenaan oli selvittää, onko näillä menetelmillä jotain yhteistä. Tapaamisen tuloksena he päättivät nimittää kevyitä menetelmiä jatkossa ketteriksi menetelmiksi (Agile Methods) ja julkaisivat näitä menetelmiä yhdistävät arvot ja niihin liittyvät 12 periaatetta. Tätä julkaisua kutsutaan ketterän ohjelmistokehityksen manifestiksi (Manifesto for Agile Software Development) ja sitä pidetään ketterien menetelmien perusmääritelmänä. (Cockburn 2007, 369–370.) Koska manifestille ei ole tehty virallista käännöstä suomeksi, se on lainattu tähän alkuperäiskielellään englannilla:

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more. (Beck, Beedle, van Bennekum, Cockburn, Cunningham, Fowler, Grenning, Highsmith, Hunt, Jeffries, Kern, Marick, Martin, Mellor, Schwaber, Sutherland & Thomas 2001a, hakupäivä 11.3.2011.)

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity--the art of maximizing the amount of work not done--is essential.

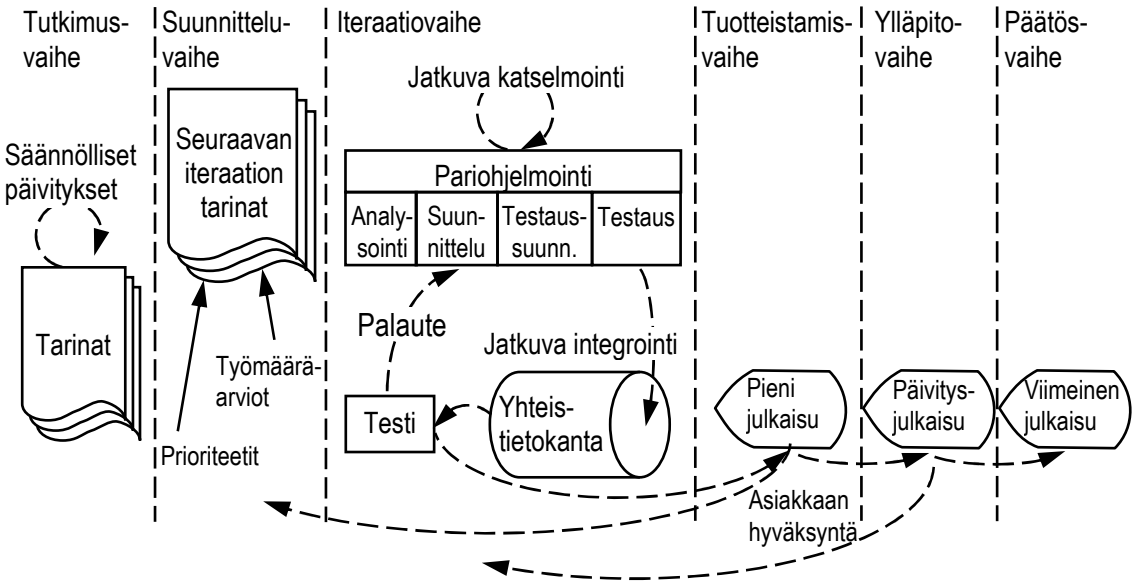
The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.
(Beck, Beedle, van Bennekum, Cockburn, Cunningham, Fowler, Grenning, Highsmith, Hunt, Jeffries, Kern, Marick, Martin, Mellor, Schwaber, Sutherland & Thomas 2001b, hakupäivä 11.3.2011.)

Abrahamsson ym. kuvaavat julkaisussaan viisi ketteriä menetelmiä yhdistävää keskeistä arvoa. Ensinnäkin ketterissä menetelmissä korostetaan ohjelmistokehittäjien välisiä suhteita ja yhteisöllisyyttä sekä inhimillisten tekijöiden huomioon ottamista sopimuksia neuvoteltaessa. Käytännössä näillä asioilla tarkoitetaan tiimin jäsenten läheisiä suhteita, tiiviitä työyhteisöjä ja tiimihenkeä parantavia menetelmiä. Toiseksi ketteryydellä pyritään siihen, että ohjelmistokehitystiimi tuottaa tasaisin väliajoin testattuja, toimivia ohjelmiston osia. Kehittäjiä kannustetaan kirjoittamaan yksinkertaista, suoraviivaista ja teknisesti edistynyttä koodia, jolloin dokumentaation määrä voidaan pitää kohtuullisena. Kolmas yhdistävä tekijä on, että kehittäjien ja asiakkaiden välistä suhdetta ja yhteistyötä suositaan enemmän kuin tiukkoja sopimuksia. Liiketoiminnallisesta näkökulmasta ketterässä ohjelmistokehityksessä keskitytään tuottamaan liiketoiminnallista arvoa välittömästi projektin alkaessa. Neljäntenä ketteriä menetelmiä yhdistävänä kohtana Abrahamsson ym. nostavat esille sekä ohjelmistokehittäjistä että asiakkaan edustajista koostuvan pätevän ja toimintavaltaisen kehitystiimin. Toimintavaltaisuudella tarkoitetaan sitä, että projektin osanottajat ovat valmiita tekemään muutoksia. Lisäksi olemassa olevien sopimusten tulee olla sellaisia, että tiimin haluamat lisäykset ja parannukset voidaan toteuttaa. (2002, 11–12.)

2.2.1 Extreme Programming

Ketteriin menetelmiin kuuluva Extreme Programming, eli lyhyemmin XP, yhdistää useita jo pitkään olemassa olleita ohjelmistokehityksen menetelmiä (Beck 1999, 9). Se on saanut vaikutteita esimerkiksi spiraalimallista ja lisäksi siihen on otettu vaikutteita eri tutkijoiden ja menetelmien kehittäjien 1980–90-luvuilla esittämistä menetelmistä (Abrahamsson ym. 2002, 22–23). Ensimmäinen tätä menetelmää noudattava projekti aloitettiin maaliskuussa 1996 (Wells 2009a, hakupäivä 14.3.2011). Extreme Programmingin mukainen prosessi jakaantuu kuuteen vaiheeseen: tutkimukseen, suunnitteluun, iteraatioon, tuotteistamiseen, ylläpitoon ja projektin päätökseen (Abrahamsson ym. 2002, 19). Tarkemmin XP:n toiminta esitetään kaaviomuodossa kuviossa 5.



KUVIO 5. Extreme Programming -menetelmän toimintaa kuvaava kaavio (Abrahamsson ym. 2002, 19, termit käännetty suomeksi)

Extreme Programming korostaa tiimityön merkitystä: sekä johtajat, asiakkaan edustajat että kehittäjät ovat tasavertaisia kumppaneita kehitystiimissä. Tämän itseohjautuvan tiimin tavoitteena on ratkaista esiin tulevat ongelmat niin tehokkaasti kuin mahdollista. (Wells 2009a, hakupäivä 14.3.2011.)

XP-menetelmässä ohjelmistoprojekteja kehitetään kiinnittämällä huomiota erityisesti kommunikaatioon, yksinkertaisuuteen, palautteeseen, kunnioitukseen ja rohkeuteen. Asiakkaiden ja ohjelmistokehittäjien välinen sekä ohjelmistokehittäjien keskinäinen kommunikaatio on projektin onnistumisen kannalta olennainen tekijä. Päivittäisestä kanssakäymisestä huolehditaan, jotta projektin lopputulokseen saadaan parhaat mahdolliset ratkaisut. Toteutettavat suunnitelmat ja koodi pyritään pitämään mahdollisimman yksinkertaisina, ja projektin aikana toteutetaan vain aidosti tarpeelliset, ohjelmiston arvoa nostavat toiminnot. (Wells 2009a, hakupäivä 14.3.2011; Wells 2009b, hakupäivä 14.3.2011.)

Kehitystiimin tulee huolehtia riittävästä palautteen saamisesta testaamalla ohjelmistoa ensimmäisestä päivästä lähtien. Asiakkaalle toimitetaan toimivia ohjelmiston osia niin aikaisessa vaiheessa kuin mahdollista, ja tarvittavat muutokset toteutetaan saadun palautteen perusteella. Sekä kehittäjien että asiakkaiden tulee kunnioittaa toistensa ammattitaitoa, ja johdon tulee luottaa kehitystiimin itseohjautuvuuteen. Kehitystiimissä pyritään siihen, että jokainen pienikin onnistuminen syventää kunnioitusta toisten tiimin jäsenten yksilöllistä osaamista ja panostusta kohtaan. Edellä

kerrottujen arvojen myötävaikutuksella kehitystiimin jäsenet voivat vastata rohkeasti muuttuviin vaatimuksiin ja teknologioihin. Projektin etenemisestä kerrotaan rehellisesti, eikä ketään tiimin jäsentä tule jättää ratkaisemaan ongelmia yksin. (Wells 2009a, hakupäivä 14.3.2011; Wells 2009b, hakupäivä 14.3.2011.)

Extreme Programming -menetelmällä toteutettaviin projekteihin kuuluu 12 käytäntöä. Projektin suunnittelu toteutetaan *suunnittelupelillä* (the Planning Game), jolloin määritellään nopeasti seuraavan julkaisun laajuus liiketoiminnallisten päämäärien ja teknisten arvioiden avulla. Tuotetta kehitetään *pienten julkaisujen* (small releases) kautta eli asiakkaan käyttöön annetaan jo projektin varhaisessa vaiheessa aikaansaatu yksinkertainen, mutta toimiva osa järjestelmää, jota sitten päivitetään julkaisemalla uusia versioita vähintään kerran kuukaudessa. Kaikkea kehitystyötä ohjataan *vertauskuvien* (metaphor) avulla. Ne ovat yksinkertaisia tarinoita siitä, miten järjestelmä toimii. *Asiakkaan läsnäolosta* (on-site customer) huolehditaan siten, että asiakas kuuluu aidosti kehitystiimiin ja on kokoaikaisesti paikalla valmiina vastaamaan esiinnousseisiin kysymyksiin. Enintään *40 tunnin työviikot* (40 hour week) nähdään XP-menetelmässä erittäin tärkeänä työhyvinvoinnin ja tehokkuuden kannalta. Kehittäjien tuleekin huolehtia siitä, ettei tämä tuntimäärä ylity ainakaan kahta viikkoa peräkkäin. (Beck 1999, 47–52.)

Extreme Programmingin käytäntöihin kuuluvalla *yksinkertaisuudella* (simple design) tarkoitetaan sitä, että järjestelmästä toteutetaan vain ne ominaisuudet, joita varmasti tiedetään tarvittavan. Myös koodin, graafisen suunnittelun ja muiden toteutuksen osa-alueiden selkeyteen kiinnitetään huomiota. (Beck 1999, 47, 50.) Extreme Programmingin edellyttämä *refaktorointi* (refactoring) tarkoittaa koodin muokkaamista siten, että varsinainen toiminnallisuus ei muutu (Sininen Meteoriitti 2011, hakupäivä 26.3.2011). Ohjelmoijien tulee huolehtia vuorovaikutuksen parantamisesta, koodissa esiintyvien päällekkäisyyksien poistamisesta, koodin yksinkertaistamisesta ja sen joustavuuden parantamisesta. XP-menetelmän mukaan kaikki ohjelmointi tapahtuu *pariohjelmoinnilla* (pair programming) eli yhdellä tietokoneella työskentelee samanaikaisesti kaksi ohjelmoijaa. Kuka tahansa tiimin ohjelmoijista voi halutessaan muokata mitä koodia tahansa milloin tahansa. Tätä kutsutaan koodin *yhteisomistajuudeksi* (collective ownership). Ohjelmoinnissa käytetään yhteisesti sovittuja *koodistandardeja* (coding standards), joiden avulla koodi pysyy yhtenäisenä ja ohjelmoijien vuorovaikutus helpottuu. (Beck 1999, 47–53.)

Joka kerta, kun jokin toteutettavan ohjelmiston toiminnoista on saatu valmiiksi, tuotettu koodi integroidaan olemassa olevaan koodiin ja yhteensopivuus testataan. Tämä on *jatkuvaa integ-*

rointia (continuous integration), jota voidaan tehdä edistymisen mukaan jopa useita kertoja päivässä. *Testaus* (testing) on olennainen osa XP-projekteja. Yksikkötesteillä ja asiakkaiden tekemällä käytännön testauksella tulee varmistaa järjestelmän toimivuus ennen uusien julkaisujen toteuttamista. (Beck 1999, 47–52.)

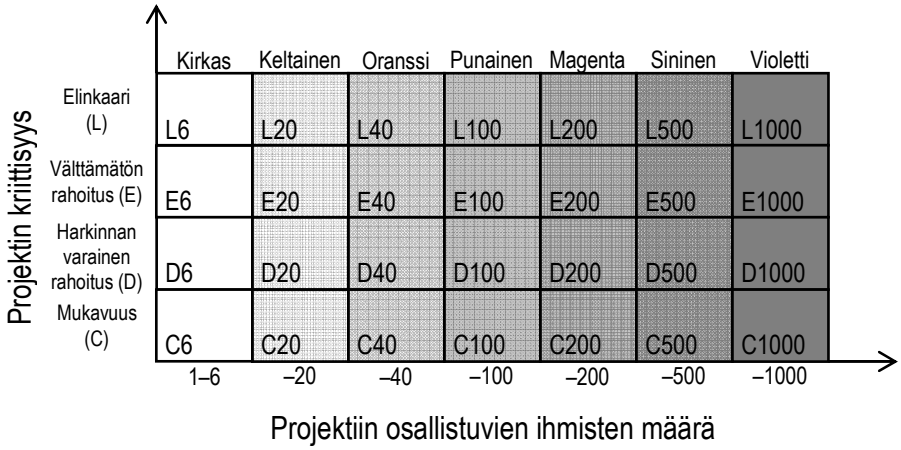
Beckin mukaan Extreme Programming ei sovellu projekteihin, joissa on monta kymmentä ohjelmistokehittäjää. Kymmenen ohjelmoijan tiimi on sopiva. Pienemmissä, kolmen tai neljän ohjelmoijan tiimeissä voidaan projektia vaarantamatta jättää joitakin käytäntöjä hyödyntämättä. XP ei sovellu käytettäväksi sellaisten teknologisten ratkaisujen kanssa, joissa ei voida huolehtia koodin yksinkertaisuudesta ja uudelleenkäytettävyydestä. Sitä ei myöskään voida hyödyntää kunnolla tilanteissa, joissa tarvitaan pitkäkestoista työtä ennen kuin toteutetusta koodista saadaan palautetta. Tällä tarkoitetaan esimerkiksi järjestelmiä, joiden kääntämiseen ja linkitykseen kuluu 24 tuntia tai joiden julkaisu vaatii usean kuukauden laadunvarmistuksen. (1999, 118.)

2.2.2 Crystal-menetelmät

Crystal-menetelmät (Crystal Methodologies) ovat Alistair Cockburnin kehittelemiä, eri tarkoituksiin sopivia ohjelmistokehitysmenetelmiä (Cockburn 2007, 337). Niiden joukosta voidaan valita sopivin metodi kullekin yksittäiselle projektille. Tämän lisäksi Crystal sisältää periaatteita, joiden avulla menetelmiä voidaan räätälöidä eri projektien vaihteleviin olosuhteisiin sopiviksi. (Abrahamsson ym. 2002, 36.)

Kuviossa 6 on esitetty Crystal-menetelmien eri vaihtoehdot taulukkomuodossa. Crystal-perheen eri jäseniä kuvataan eri väreillä, jotka ilmaisevat kyseisen menetelmän "raskautta": mitä tummempi väri, sitä raskaampi menetelmä (Abrahamsson ym. 2002, 36). Värit etenevät vaakasuunnassa kirkkaasta keltaiseen, oranssiin, punaiseen, magentaan, siniseen, violettiin ja niin edelleen. Eri menetelmiä kuvataan värin mukaisella nimityksellä, esimerkiksi nimellä Crystal Clear tai Crystal Orange. (Cockburn 2007, 338.) Värien lisäksi Crystal-menetelmien eroja kuvataan kuvion pystyakselin kirjaimilla, jotka kuvaavat projektin pettämisen yhteydessä syntyvää vahinkoa. Nämä kirjaimet merkitsevät mukavuuden menettämistä (C, loss of comfort), harkinnanvaraisen rahoituksen menettämistä (D, loss of "discretionary" moneys), välttämättömän rahoituksen menettämistä (E, loss of "essential" moneys) eli konkurssia ja koko elinkaaren, projektin elämän, menettämistä (L, life). Taulukossa kirjaimen perässä oleva numero ilmaisee projektiin osallistuvien ihmisten enimmäismäärää. Esimerkiksi Crystal Clear -menetelmän mukaisissa

projekteissa on 1–6 jäsentä ja Crystal Orangessa 21–40 osallistujaa. (Cockburn 2001, hakupäivä 14.3.2011.)



KUVIO 6. Crystal-menetelmät (Cockburn 2001, hakupäivä 14.3.2011, termit käännetty suomeksi)

Kaikki Crystal-menetelmät tarjoavat suuntaviivat, joita kehitysprosessissa tulee noudattaa. Ohjeet ottavat kantaa muun muassa menettelykäytäntöihin, eri vaiheiden tuotoksiin, työkaluihin, standardeihin ja rooleihin. (Abrahamsson ym. 2002, 38.) Oikean Crystal-menetelmän valinnassa tulee huomioida esimerkiksi projektiin osallistuvien ohjelmistokehittäjien määrä ja projektiin liittyvät riskit. Osallistujien määrän huomioiminen on tärkeää sen vuoksi, että eri menetelmissä on käytössä erilaiset keinot kommunikaation onnistumiseen. Menetelmäksi tulisi valita kevyin menetelmä, jonka voidaan ajatella toimivan kyseisessä projektissa. Tällöin toimitaan tehokkaimmin ja päästään parhaaseen lopputulokseen. (Cockburn 2001, hakupäivä 14.3.2011.)

Esimerkkinä Crystal-menetelmistä tarkastellaan pääsääntöisesti 1–6 henkilön projekteihin sopivaa Crystal Clear -menetelmää. Projektin työntekijöiden tulisi työskennellä yhdessä tiimissä joko samassa huoneessa tai vierekkäisissä huoneissa. Tiimissä on neljä eri roolia: sponsori, vanhempi suunnittelija-ohjelmoija, suunnittelija-ohjelmoija ja käyttäjä. Yksi tiimin jäsenistä voi olla projektin koordinaattori, ja lisäksi tarvitaan liiketoiminnallista osaamista. Vaatimusten määrittelyyn osallistuu yksi tai useampi tiimin jäsen. (Cockburn 2007, 340.)

Crystal Clearin käytäntöihin kuuluu, että ohjelmistoa kehitetään inkrementaalisesti ja siitä julkaistaan säännöllisesti kahden tai kolmen kuukauden välein uusia osia. Edistymistä seurataan ohjelmiston osien julkaisun tai tärkeiden päätösten perusteella. Kirjalliset dokumentit eivät ole oleellisessa asemassa, mutta niitäkään ei saa laiminlyödä kokonaan, jotta esimerkiksi mahdolliset

uudet tiimin jäsenet voivat tutustua helpommin jo toteutettuun ohjelmistoon. Crystal Clearissa hyödynnetään automaattista regressiotestausta jossain määrin. Ohjelmiston loppukäyttäjän tulee olla mukana ohjelmiston kehitystyössä ja jokaisesta julkaisusta järjestetään kaksi käyttäjäkatselmusta. Jokaisen kahden–kolmen kuukauden mittaisen kehitysjakson alussa ja puolivälissä pidetään palaveri, jossa pohditaan tuotetta ja käytetyn menetelmän onnistumista. (Cockburn 2007, 340–341.)

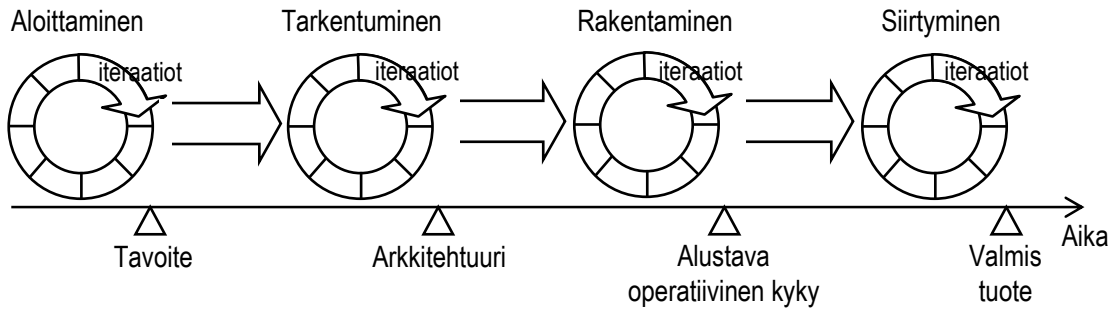
Crystal-menetelmissä ei suljeta pois muiden ohjelmistokehitysmenetelmien käyttöä. Niinpä joitain osa-alueita voidaan korvata samankaltaisten menetelmien käytännöillä. Tiimi voi esimerkiksi päättää ottaa käyttöön Scrumin päivittäiset palaverit tai Extreme Programming -menetelmän pariohjelmoinnin. (Cockburn 2007, 341.)

Verratessaan Extreme Programming - ja Crystal Clear -menetelmiä Cockburn huomioi, että XP ja Crystal Clear ovat tietyllä tavalla samankaltaisia. Hän näkee XP:n menetelmänä, jonka tiukkoja sääntöjä noudattamalla voidaan saavuttaa suuri tuottavuus. Hänen mielestään XP:n käytäntöjen noudattaminen on kuitenkin vaikeampaa kuin Crystal Clearin. Crystal Clear sallii tiimin sisällä enemmän yksilöllisyyttä ja rennompia työtapoja, mutta samalla tehokkuus kärsii hieman. Crystal Clear on helpompi omaksua, mutta jopa mallin kehittäjä itse myöntää, että Extreme Programming johtaa parempiin lopputuloksiin, mikäli kehitystiimi vaan kykenee noudattamaan sitä. (2001, hakupäivä 14.3.2011.)

2.2.3 Rational Unified Process

Rational Unified Process eli RUP on Philippe Kruchtenin, Ivar Jacobsenin ja muiden Rational Corporationin työntekijöiden kehittämä ohjelmistokehitysmenetelmä, joka täydentää UML-mallinnuskielen (Unified Modeling Language) käytäntöjä. RUP on iteratiivinen lähestymistapa olio-ohjelmoinnilla toteutettaviin järjestelmiin, ja siinä nojataan vahvasti käyttötapausten kautta hahmoteltuihin vaatimuksiin. (Abrahamsson ym. 2002, 55.)

RUP-menetelmässä projektin elinkaari jaetaan neljään vaiheeseen: aloittamiseen, tarkentumiseen, rakentamiseen ja siirtymiseen. Kukin näistä vaiheista on jaettu iteraatioihin eli kehitysjaksoihin, joista jokaisen tarkoituksena on tuottaa demonstroitava osa ohjelmistoa. (Abrahamsson ym. 2002, 55.) RUP-menetelmää havainnollistetaan kuviossa 7.



KUVIO 7. Rational Unified Process -menetelmän vaiheet (yhdistetty lähteistä Abrahamsson ym. 2002, 56 ja Rational Software Corporation 2001, hakupäivä 30.3.2011, termit suomennettu)

RUPin käytäntöjen mukaan yhden iteraation kesto voi vaihdella alle kahdesta viikosta enintään kuuteen kuukauteen (Abrahamsson ym. 2002, 55). Jokainen neljästä vaiheesta päättyy rajapyykkiin, jonka onnistumisen perusteella päätetään seuraavaan vaiheeseen siirtymisestä. Kuhunkin vaiheeseen käytettävä aika ja vaiva vaihtelevat. Ymmärrettävästi eniten resursseja tarvitaan järjestelmän rakentamisvaiheessa. Karkeasti arvioituna aloitusvaiheeseen menee 10 prosenttia projektin kokonaisajasta, tarkentumiseen 30 prosenttia, rakentamiseen eniten eli 50 prosenttia ja siirtymisvaiheeseen jäljelle jääneet 10 prosenttia. (Rational Software Corporation 2001, hakupäivä 30.3.2011.)

Rational Unified Process -menetelmässä on koottu yhteen kaupallisesti toimiviksi todettuja käytäntöjä. Nämä käytännöt ovat ohjelmiston kehittäminen iteratiivisesti, vaatimusten hallinnoiminen, komponenttipohjaisen arkkitehtuurin käyttäminen, ohjelmiston visuaalinen mallintaminen, ohjelmiston laadun varmistaminen ja ohjelmistoon tehtyjen muutosten hallitseminen. (Rational Software Corporation 2001, hakupäivä 30.3.2011.)

Iteratiivisen kehittämisen avulla pienennetään projektiin liittyvää riskiä. Tähän päästään tuottamalla säännöllisin väliajoin toiminnallisia julkaisuja. Niiden avulla loppukäyttäjä sitoutetaan projektiin ja häneltä saadaan palautetta. Vaatimusten hallinnoimisella tarkoitetaan systemaattista lähestymistapaa muuttuvien vaatimusten etsimiseen, dokumentointiin, organisointiin ja jäljittämiseen. (Rational Software Corporation 2001, hakupäivä 30.3.2011.)

RUP-menetelmän mukaisesti toteutettavan ohjelmiston arkkitehtuurin tulee pohjautua ohjelmistokomponentteihin (Rational Software Corporation 2001, hakupäivä 30.3.2011). Niihin pohjautuvalla ohjelmistoarkkitehtuurilla tarkoitetaan itsenäisistä, erikseen kehitettävistä ja yhteen koottavista koodiosioista rakentuvaa ohjelmistoa. Ohjelmistokomponentit ovat uudelleenkäytettäviä ja

suoritettavissa olevia ohjelmiston osia, joita käytetään komponentin tarjoamien rajapintojen kautta. (Kuikka 2011, hakupäivä 30.3.2011.)

Rational Unified Process -menetelmän mukaisesti toteutettavaa järjestelmää mallinnetaan UML:ää (Unified Modelling Language) hyödyntämällä. Tällä pyritään monimutkaisen järjestelmän ymmärtämisen edistämiseen, kehitystiimin kommunikaation parantamiseen, suunnitelma-vaihtoehtojen kartoittamiseen ja vertaamiseen sekä vaatimusten täsmälliseen tallentamiseen. (Rational Software Corporation 2001, hakupäivä 30.3.2011.)

Projektin aikana julkaistavien tuotosten laadun arvioimista tulee RUP-menetelmän mukaan tehdä useita kertoja. Tämä voidaan toteuttaa kunkin iteraation tärkeimpien skenaarioiden demonstraatiolla ja testauksella. Muutoksen hallinnoiminen on avainasemassa ohjelmistokehityksessä, erityisesti kun mukana on useita kehittäjiä mahdollisesti useaan tiimiin jaettuina. RUP kiinnittääkin huomiota esimerkiksi ohjelmistoon tehtävien muutosten hallinnoimiseen. (Rational Software Corporation 2001, hakupäivä 30.3.2011.)

Rational Unified Processin käyttöönotto vaatii usein menetelmän muokkaamista juuri kyseiseen yritykseen ja prosessiin sopivaksi. Tästä laajamittaisen räätälöinnin tarpeesta huolimatta RUP on käytössä monissa organisaatioissa. Osaltaan tähän on vaikuttanut se, että menetelmän oikeudet omistava Rational Software myy suosittuja työkaluja, jotka tukevat RUPin eri vaiheita. Tällaisia työkaluja ovat esimerkiksi UML-työkalu Rational Rose ja ohjelmistokokoonpanon hallintatyökalu ClearCase. (Abrahamsson ym. 2002, 60.)

3 SCRUM

Ketteriin ohjelmistokehitysmenetelmiin kuuluva Scrum kokoaa yhteen erilaisia ohjelmistokehitysprojektien hallintaan liittyviä teknisiä käytäntöjä (Leffingwell & Smits 2005, 3–4). Englanninkielisissä lähteissä siitä käytetäänkin usein menetelmän (method) sijasta termiä "framework". Se ei siis ole tuotteiden valmistukseen käytettävä yksittäinen prosessi tai tekniikka, vaan sen avulla voidaan nostaa esille ja parantaa yrityksen tuotekehityksen tehokkuuteen vaikuttavia tekijöitä (Schwaber & Sutherland 2010, 3).

Scrum soveltuu käytettäväksi iteratiivisissa prosesseissa, jotka kehitysjakson päätteeksi tuottavat julkaisukelpoista toiminnallisuutta. Se sopii erityisesti tiimipohjaisiin kehitysprojekteihin, joiden vaatimukset voivat muuttua nopeasti. Scrumin avulla voidaan havaita ja poistaa sellaisia haittatekijöitä, jotka voivat vaarantaa kehitystyön ja kehitettävän tuotteen. Sen käytöllä tähdätään erityisesti tuottavuuden maksimoimiseen sekä kommunikaation ja yhteistyön parantamiseen. (Leffingwell & Smits 2005, 4.)

Scrum ei ole lyhenne vaan termi, joka juontaa juurensa rugbyyn liittyvään sanastoon. Rygbyssä sillä tarkoitetaan strategiaa, jota joukkueet käyttävät kamppaillessaan pelikentälle palautettavasta pallosta. Ikujiro Nonaka ja Hirotaka Takeuchi lainasivat vuonna 1987 tämän termin kuvaamaan Japanissa käytettyä tuotekehitysprosessia. Sekä rugby-palloilulaji että ketteriin ohjelmistokehitystapoihin kuuluva Scrum ovat mukautuvia, nopeita ja itseohjautuvia, eikä kummassakaan pidetä pitkiä taukoja etenemisen välillä. (Schwaber & Beedle 2001, 1.)

Kirjassaan Agile Project Management with Scrum Schwaber kertoo Scrumin pohjautuvan empiirisen prosessinhallinnan teorioihin. Nämä teoriat nojaavat kolmeen tukipilariin: läpinäkyvyyteen, tarkasteluun ja sopeuttamiseen. *Läpinäkyvyydellä* varmistetaan, että kaikki prosessin lopputulokseen vaikuttavat tekijät ovat lopputulosta hallinnoivien henkilöiden tiedossa, ja että he tuntevat nämä tekijät läpikotaisesti. Toistuvan *tarkastelun* avulla ei-hyväksyttävät muutokset voidaan havaita, ja niihin voidaan reagoida riittävän ajoissa. Mikäli tarkastelussa huomataan, että yksi tai useampi projektin osa-alueista on hyväksyttävien rajojen ulkopuolella ja projektin lopputuloksesta uhkaa tulla ei-hyväksyttävä, täytyy prosessia *sopeuttaa* mahdollisimman nopeasti. Tällä estetään projektia vinoutumasta edelleen. (2004, 3–4.)

Koko Scrum-prosessia voidaan kuvata lyhyesti seuraavan esimerkin avulla. Ensiksi johto valitsee muutamia ohjelmoijia ja antaa heille työtilat sekä projektin toteutettavaksi. Aluksi tiimin jäsenet keskustelevat asiakkaan kanssa selvittääkseen, millaisen tuotteen hän haluaa ja mikä on hänelle tärkeää. Tätä palaveria nimitetään Scrumissa julkaisun suunnittelukokoukseksi. Tiimiläiset laativat yhdessä asiakkaan kanssa tärkeysjärjestykseen laitetun ominaisuuslistan eli tuotteen työlistan. Pannessaan kehitystyön alulle, tiimi valitsee yhdessä asiakkaan kanssa muutamia ominaisuuksia, jotka toteutetaan ensimmäisessä sprintissä tärkeytensä mukaisesti. Tämä tapaaminen on sprintin suunnittelukokous ja valitut ominaisuudet muodostavat sprintin tehtävälistan. (Schwaber & Beedle 2001, 21.) Sprintiksi Scrumissa kutsutaan enintään kuukauden mittaista aikajaksoa, jolloin ohjelmistokehitystyötä tehdään. Yhdessä kehitysprojektissa on useita sprinttejä ja jokaisen sprintin lopputuloksena on mahdollisesti julkaisukelpoinen osa valmista tuotetta. (Schwaber & Sutherland 2010, 4–5.) Työn edetessä sprintin tehtävälistaa täydennetään niillä ongelmilla, joita tiimi kohtaa tuotetta kehittäessään. Pysyäkseen tietoisina siitä, mitä kulloinkin on tehty ja mitä toteutetaan seuraavaksi, tiimi pitää vapaamuotoisia palavereita, joissa kukin kertoo mitä on tehnyt ja mitä aikoo tehdä seuraavaksi. Näitä kokouksia kutsutaan nimellä Päivän Scrum. Kun ominaisuuksia on saatu toteutettua, tiimin jäsenet esittävät aikaansaannokset johdolle ja asiakkaalle sprintin katselmointipalaverissa. (Schwaber & Beedle 2001, 21.)

3.1 Scrum-sanasto

Scrumiin liittyy useita termejä, jotka on kehitetty varta vasten tätä ketterää menetelmää varten. Taulukossa 1 esitetyn sanaston laatimisessa on hyödynnetty soveltuvin osin Szalvayn (2007, hakupäivä 30.3.2011) julkaisemaa Scrum-sanastoa, Lindströmin (2006, hakupäivä 30.3.2011) laatimaa suomenkielistä sanastoa sekä Schwaberin ja Sutherlandin (2010) kirjoittamaa Scrum-opasta.

TAULUKKO 1. Scrum-sanastoa

Termi (suluissa englanniksi)	Termin selitys
Scrum-mestari (ScrumMaster)	Scrumilla toteutettavan projektin tehokasta etenemistä edistävä henkilö, joka avustaa kehitystiimin jäseniä ja tuoteomistajaa heidän omien tehtäviensä hoitamisessa.
Tuoteomistaja (Product Owner)	Henkilö, joka hallinnoi Scrum-projektia ja vastaa siitä, että kehitystiimi toteuttaa liiketoiminnan kannalta tärkeimpiä asioita.
Kehitystiimi (Team)	Ohjelmistokehitysammattilaista koostuva itseohjautuva tiimi, joka suunnittelee ja toteuttaa käytännössä Scrum-projektissa tehtävän ohjelmiston. (Kun tässä opinnäytetyössä käytetään termiä "tiimi", tarkoitetaan kehitystiimiä, ei Scrum-tiimiä.)
Scrum-tiimi (Scrum Team)	Scrum-mestari, tuoteomistaja ja kehitystiimi
Päivän Scrum (Daily Scrum Meeting)	Kehitystiimin ja Scrum-mestarin pitämä 5–15 minuutin mittainen päivittäinen tiimin tilannekatsaus.
Julkaisun suunnittelukokous (Release Planning Meeting)	Palaveri, jossa suunnitellaan toteutettava julkaisu eli lopputuote ja määritellään sille tavoitteet.
Tuotteen työlista (Product Backlog)	Dokumentti, joka sisältää toteutettavalle järjestelmälle asetetut vaatimukset ja toiminnallisuudet työmääräarvioineen tärkeysjärjestyksessä.
Julkaisun edistymiskäyrä (Release Burndown Chart)	Kaavio, jonka avulla voidaan seurata toteutettavan julkaisun eli lopputuotteen edistymistä. Kaavion avulla havainnollistetaan jäljellä olevien ominaisuuksien työmäärää suhteessa käytettävissä olevaan aikaan.

Sprintti (Sprint)	Yleensä enintään 4 viikon mittainen kehitysjakso, iteraatio, jonka aikana etukäteen sovitut tuotteen ominaisuudet toteutetaan. Sprinttejä järjestetään peräkkäin aina Scrum-projektin päättymiseen asti.
Sprintin suunnittelukokous (Sprint Planning Meeting)	Palaveri, jossa kehitystiimi ja tuoteomistaja neuvottelevat siitä, mitä julkaistavan tuotteen ominaisuuksia tiimi toteuttaa seuraavan sprintin aikana.
Sprintin tehtävälista (Sprint Backlog)	Dokumentti, jossa määritellään kyseisessä sprintissä toteutettavat toiminnot ja ominaisuudet tarkalla tasolla.
Sprintin edistymiskäyrä (Sprint Burndown Chart)	Kaavio, joka havainnollistaa päivätasolla sprintin tehtävälistan jäljellä olevien tehtävien työmäärää suhteessa käytettävissä olevaan aikaan.
Sprintin katselmointi (Sprint Review)	Palaveri, jossa esitellään lyhyesti kehitystiimin kyseisen sprintin aikana toteuttama ohjelmiston osa ja keskustellaan avoimesti sprintin kulusta.
Sprintin jälkitarkastelu (Sprint Retrospective Meeting)	Palaveri, joka järjestetään jokaisen sprintin päätteeksi ja jossa Scrum-mestari ja kehitystiimi keskustelevat, mikä päättyvässä sprintissä meni hyvin ja mitä parannuksia seuraavaa sprinttiä varten tehdään.

3.2 Arvot

Schwaberin ja Beedlen teoksessa Agile Software Development with Scrum kuvataan Scrumin pohjautuvan viiteen perusarvoon. Nämä ovat sitoutuminen, keskittyminen, avoimuus, kunnioitus ja rohkeus. (2001, 147.)

Sitoutumisella tarkoitetaan halukkuutta sitoutua määriteltyihin tavoitteisiin. Tiimin jäsenten sitoutumista auttaa se, että heille annetaan oikeus tehdä itsenäisiä päätöksiä. Tämä ei kuitenkaan onnistu, mikäli tiimiläiset eivät tiedä, mitä heidän tulee tarkalleen ottaen tehdä. (Schwaber &

Beedle 2001, 148.) Tämän vuoksi on tärkeää, että sprintin tehtävälistaan (katso luku 3.5.3) kirjatut ominaisuudet ja toiminnot on määritelty selkeästi.

Kunkin tiimin jäsenen tulee *keskittyä* kulloinkin vastuualueenaan olevaan tehtävään. Mihinkään muuhun, tehtävään kuulumattomaan asiaan ei pidä kiinnittää huomiota. Työssä käytetyn ajan pitäisi kohdistua yhden ongelman ratkaisemiseen kerrallaan. (Schwaber & Beedle 2001, 149.)

Avoimuus on edellytys sille, että Scrum yleensäkin toimii ja tekee sen, mitä sen odotetaan tekevän. On tärkeää, että projektin todellinen tilanne on koko ajan selvillä, jotta ei jouduta tekemään aikaa vieviä korjauksia aikaisemmin toteutettuihin tehtäviin tai toteamaan pitkän työrupeaman jälkeen, ettei jostain syystä saatukaan aikaiseksi toivottua lopputulosta. Scrumissa avoimuudesta huolehditaan esimerkiksi siten, että tuotteen työlista (katso luku 3.5.1) on kaikkien nähtävillä. Päivittäisten Scrum-palavereiden (katso luku 3.4.4) avulla kaikki osalliset tietävät, mitä kulloinkin on työn alla. Sprintin lopputulos esitellään avoimesti sprintin katselmointikokouksessa (katso luku 3.4.5). Koska Scrumissa seurataan tekemättömän työn määrää suhteessa käytettävissä olevaan aikaan, projektin kehityssuunta ja nopeus ovat jatkuvasti hallittavissa. (Schwaber & Beedle 2001, 151.)

On tärkeää, että tiimiläiset *kunnioittavat* toisiaan. Johto valitsee jokaisen tiimin työntekijän, ja Scrum muodostaa ympäristön, jossa ihmiset työskentelevät tiiminä. Jokaisella tiimin jäsenellä on omanlaisensa tausta, koulutus ja työkokemus, omat vahvuutensa ja heikkoutensa. Scrumin tarkoituksena on, että yksilöt yhdistävät taitonsa ja muodostavat keskinäiseen kunnioitukseen nojaavan tiimin. Tämä voi vaatia Scrum-mestarin ohjausta. Muutaman yhteistyönä tehdyn sprintin jälkeen tiimiläisten tulisi tuntea toistensa vahvuudet ja heikkoudet, hyväksyä ne ja oppia kuinka mukauttaa toimintaa niiden mukaisesti. (Schwaber & Beedle 2001, 152–153.)

Rohkeudella tarkoitetaan uskallusta toteuttaa projektia Scrumin mukaisesti. Monet organisaatiot toimivat edelleen hierarkkisesti ja autoritaarisesti, joten uuden menetelmän käyttäminen voi vaatia kanttia. Lisäksi rohkeudella tarkoitetaan kykyä luottaa itseensä ja siihen, että oman harkintakyvyn käyttö on kunnioitettavaa. Se on itsepäisyyttä, periksiantamattomuutta ja sisua. Rohkeutta on myös se, että kertoo avoimesti tarvitsevansa apua muilta tiimin jäseniltä. (Schwaber & Beedle 2001, 153–154.)

3.3 Scrum-tiimi ja sen roolit

Scrumin periaatteiden mukaisesti toteutettavissa projekteissa työ tehdään Scrum-tiimissä. Jokaisessa Scrum-tiimissä on kolme roolia. *Scrum-mestarin* vastuulla on varmistaa, että prosessinkulku on ymmärretty oikein ja että sitä noudatetaan. *Tuoteomistaja* huolehtii siitä, että Scrum-tiimin työn arvo on paras mahdollinen. *Kehitystiimin* jäsenet puolestaan tekevät varsinaisen ohjelmistokehitystyön. Kehitystiimin jäseniä valittaessa on pidettävä mielessä, että tiimistä on löydyttävä kaikki tarvittava tietämys projektin läpiviemiseksi ja halutun lopputuloksen aikaansaamiseksi. (Schwaber & Sutherland 2010, 4.)

Kaikki Scrum-tiimin jäsenet vastaavat projektin lopputuloksesta oman osaamisensa kautta. Scrum-mestarin pitää huolehtia, että kehitystiimi selviytyy tehtävistään: jos kehitystiimi ei onnistu tuottamaan toivottua lopputulosta, Scrum-mestari ei ole hoitanut tehtäväänsä riittävän hyvin. (Virtanen 17.6.2010, luento.) Kehitystiimi vastaa varsinaisesta kehitystyöstä, mutta projektin johdon tulee tukea heitä tehtävässään poistamalla tehokasta työskentelyä hidastavat esteet. Tehostamalla kehitystiimin toimintaa haittatekijöihin puuttumalla, Scrum-mestari voi vaikuttaa projektin liiketoiminnalliseen tehokkuuteen, siihen että kehitystiimi saa tuotettua liiketoiminnallista arvoa. (Leffingwell & Smits 2005, 6.)

Scrum-tiimiin kuuluvista jäsenistä käytetään nimitystä "siat". Kaikki tiimin ulkopuoliset ihmiset ovat "kanoja". Kanoilla ei ole oikeutta kertoa sioille, kuinka näiden tulisi tehdä työnsä. Nimitykset "sika" ja "kana" perustuvat seuraavaan tarinaan: "Kanan ja sian tavatessa kana ehdottaa: 'Perustetaanko yhdessä ravintola?' Sika ajattelee asiaa ja kysyy: 'Minkä nimen antaisimme tälle ravintolalle?' Kana sanoo: 'Pekonia ja munia!' Tämän kuultuaan sika toteaa: 'Ei kiitos, sillä minä joutuisin sitoutumaan, mutta sinä olisit vain mukana!'" (Schwaber & Sutherland 2010, 5–6.)

"Siat" – Scrum-mestari, tuoteomistaja ja kehitystiimi – ovat Scrum-projektin johtavassa roolissa. Ajatus siitä, että kehitystiimi johtaa itse itseään voi vaikuttaa oudolta. Tämä on kuitenkin yksi Scrumin keskeisimpiä opinkappaleita. Kaikki Scrum-tiimin ulkopuoliset organisaation johtajat kuuluvat "kanoihin". He saattavat olla kiinnostuneita projektista ja sen onnistumisesta, mutta heidän ainoa mahdollisuutensa ottaa osaa projektiin on toimia "sikojen" kautta. "Kanoilla" ei ole suoraa käskyvaltaa siihen, miten projektia pitäisi toteuttaa. (Schwaber 2004, 15.)

3.3.1 Scrum-mestari

Scrum-mestari (ScrumMaster) vastaa Scrum-prosessin käytänteiden oikeaoppisesta hyödyntämisestä projektissa (Virtanen 17.6.2010, luento). Tämä ei ole helppo tehtävä vaan vaatii asioihin perehtymistä, kokemusta ja paneutumista. Periaatteessa Scrum-mestarin nimikkeenä voidaan käyttää myös projektipäällikköä, mutta Schwaber (2004, 25) kertoo kehittäneensä uuden nimikkeen, koska hän halusi korostaa Scrum-mestarin tehtävien eroamista perinteisistä projektipäällikön tehtävistä. Deemer, Benefield, Larman ja Vodde (2010, 6) korostavat, että Scrum-mestari ei ole tiimin johtaja, vaan hänen roolinsa on palveleva.

Scrum-mestarin päätehtävänä on huolehtia, että projektia viedään eteenpäin Scrumin käytäntöjen, arvojen ja sääntöjen mukaisesti (Schwaber & Sutherland 2010, 6). Hän on vastuussa mahdollisten kehitystiimin, tuoteomistajan ja asiakkaan välisten esteiden poistamisesta. Yksi esteistä voi olla kehitystiimin ja asiakkaan edustajien käyttämä erilainen kieli. IT-alan ammattilaiset ovat tottuneet puhumaan teknistä kieltä, kun taas tuoteomistaja voi käyttää liiketoimintaan liittyviä termejä. Kyseessä on tilanne, jossa Scrumia käyttävän yrityksen tulee tarjota hyvää palvelua asiakkaalle. Näin ollen Scrum-mestarin tulee opastaa kehitystiimiä käyttämään tuoteomistajalle ja muille asiakkaan edustajille tuttua kieltä. Tämä käy helpommin kuin teknisen termistön opettaminen ihmiselle, jolla ei ole IT-puolen taustaa. (Schwaber, 2004, 53, 65–66.)

Kehitystiimin tulee ymmärtää ja osata käyttää hyödykseen itseohjautumista sekä tiimin jäsenten monipuolista ammattiosaamista. Tämän tietämyksen kasvattamisessa Scrum-mestari on avainasemassa. (Schwaber & Sutherland 2010, 6.) Hän auttaa tiimiä myös Scrumin mukaisessa päätöksenteossa, huolehtii, että projektilla on käytössään tarvittavat resurssit ja ettei mikään häiritse tiimin tehokasta toimintaa (Schwaber & Beedle 2001, 8). Pienissä kehitystiimeissä joku tiimin jäsenistä voi hoitaa muiden tehtävien ohella Scrum-mestarin tehtäviä. Tällöin hänellä täytyy olla riittävästi aikaa Scrum-mestarina toimimiseen eli vähemmän kehitystyötä kuin muilla tiimin jäsenillä. (Deemer ym. 2010, 6.)

Kehitystiimin opastamisen lisäksi myös tuoteomistajan opastaminen tehtäviensä hoitamisessa kuuluu Scrum-mestarin vastuualueeseen (Schwaber & Sutherland 2010, 6). Scrum-mestari voi joutua puuttumaan tuoteomistajan tekemisiin esimerkiksi, jos tämä haluaa lisätä kehitystiimin tehtäviä kesken sprintin. Tällaisten seikkojen vuoksi Scrum-mestari ei koskaan saa toimia tuoteomistajana. (Deemer ym. 2010, 7.) Scrum-mestari auttaa tuoteomistajaa valitsemaan tärkeimmät

ja arvoltaan suurimmat kohdat tuotteen työlistalta, vaikka tuoteomistaja onkin se, joka viimekädessä vastaa tärkeysjärjestyksestä. Mikäli tuoteomistaja ei ole aiheeseen perehtynyt, Scrum-mestari voi näyttää hänelle, kuinka Scrumia hyödynnetään sijoitetun pääoman tuoton maksimoimisessa ja projektin tavoitteiden saavuttamisessa. (Schwaber 2004, 15, 53.)

Scrum-mestari ei opasta kehitystiimiä ja tuoteomistajaa sekä muuta yrityksen organisaatiota pelkästään Scrumin periaatteiden omaksumisessa. Hän myös auttaa Scrum-tiimiä tekemään työnsä mahdollisimman hyvin tilanteessa, jossa organisaatio ei toimi optimaalisesti monimutkaisen tuotekehityksen kannalta. Tämän valmentamisen ja johtamisen tavoitteena on tuottavuuden kasvattaminen ja korkealaatuisten tuotteiden valmistuminen. (Schwaber & Sutherland 2010, 6.)

Schwaberin mukaan Scrum-mestarin vastuut ja velvollisuudet voidaan tiivistää kuuteen asiaan. Hänen pitää poistaa kehityksen ja tuoteomistajan väliset esteet, jotta tuoteomistaja voi ohjata kehitystä suoraan. Lisäksi hänen pitää opettaa tuoteomistajaa tuotteeseen sijoitetun pääoman maksimoimisessa ja siinä, miten tämä voi saavuttaa päämääränsä Scrumin avulla. Scrum-mestarin tulee huolehtia kehitystiimin toiminnan parantamisesta luovuutta ja täysivaltaisuutta helpottamalla. Hänen täytyy myös parantaa kehitystiimin tuottavuutta kaikin mahdollisin tavoin. Hänen tulee kohentaa tuotteen kehityksen käytäntöjä ja työkaluja siten, että jokainen toteutettava toiminnallisuus on julkaistavissa. Lisäksi hänen vastuunaan on tiedottaa tiimin kehityksestä avoimesti ja ajantasaisesti kaikille osapuolille. Kun nämä velvollisuudet hoidetaan vastuullisesti, projekti pysyy hallinnassa. (2004, 36.)

Adkins kirjoittaa teoksessaan Coaching Agile Teams Scrum-mestarin erilaisista valmentamiseen liittyvistä rooleista, jotka eroavat suurelta osin perinteisistä projektipäällikön tehtävistä. Hän erittelee kuusi roolia: valmentaja-mentori (coach-mentor), fasilitaattori (facilitator), opettaja (teacher), ongelmanratkaisija (problem solver), konfliktien selvittäjä (conflict navigator) ja yhteistyön johdattelija (collaboration conductor). (2010, xxiii–xxiv.)

Valmentaja-mentorin rooliin liittyy sekä yksittäisten kehitystiimin jäsenten, tuoteomistajan että koko Scrum-tiimin valmentaminen. Mentorointia tehdään projektin jokaisessa vaiheessa, mutta erityisen aktiivista valmentamista tarvitaan sprintin alussa ja lopussa. Varsinkin silloin, kun tiimi on aloittamassa ensimmäistä yhteistä Scrum-projektiaan, on hyvä järjestää koulutus, johon kaikki Scrum-tiimin jäsenet osallistuvat. Sprintin puolivälissä Scrum-mestari voi järjestää yhteisen

tapaamisen, jossa käydään kevyellä tasolla läpi, miten sprintti saatetaan loppuun. Muuten puolivälissä keskitytään yksilölliseen valmentamiseen. Valmentaja-mentorin rooliin kuuluu myös, että hän kannustaa tiimiläisiä ratkaisemaan kahdenvälisiä ongelmia asianosaisten kesken. Sprintin loppupuoli on luonnollinen hetki arvioida koko tiimin kesken sitä, miten yhteistyö onnistui ja pohtia, miten tiimin toimintaa voidaan parantaa entisestään seuraavaa sprinttiä ajatellen. Koska sprintit seuraavat jatkuvassa syötössä toisiaan, voidaan sprintin alun ja lopun mentorointi hoitaa usein yhden kokoontumisen aikana. (Adkins 2010, 78–81.)

Scrumin perustana on itseohjautuva kehitystiimi. Tämä ei kuitenkaan tarkoita sitä, etteikö Scrum-mestari saisi omalla panoksellaan osallistua tiimin toimintaan kohdattaessa ongelmatilanteita. Adkinsin mukaan *fasilitointia* tehdään kaikissa Scrum-projektiin kuuluvissa palavereissa ja tapaamisissa. Fasilitaattori huolehtii siitä, että kehitystiimissä esiin nousevat ideat ja keksinnöt kirjataan ylös. Tähän liittyy Scrum-mestarille kuuluva tehtävä tukea tiimin itseohjautumista ja parantaa heidän kykyään tuottaa toiminnallaan todellista liiketoiminnallista arvoa. Jotta itseohjautuminen on mahdollista, fasilitointia tulee tehdä kevyellä otteella. Tämä tarkoittaa sitä, että Scrumiin kuuluvissa tapaamisissa pitää asetut tavoitteet pyrkiä saavuttamaan ilman, että Scrum-mestari sortuu johtamaan tiimiä. Hänen täytyykin luottaa tiimin jäseniin ja muistuttaa itselleen, että "This is their meeting, not mine". (2010, 118–119, 143.)

Scrum-mestari on *opettaja*-roolissa lukuisia kertoja koko Scrum-tiimin olemassaolon ajan. Erityisesti vastamuodostetulla tiimillä on paljon opittavaa ketteryydestä, Scrumista, siihen liittyvistä rooleista ja niiden vaatimuksista. Opettamista tarvitaan, kun tiimiin liittyy uusi jäsen, vaikka tämä olisi aikaisemminkin työskennellyt Scrumin mukaisesti. Scrum-mestarin tulee tarkkailla tilanteita ja toimia opettajan roolissa, kun hän huomaa tiimin tai sen yksittäisten jäsenten tarvitsevan opastusta. Opettajan tulee huolehtia siitä, että Scrum-tiimi arvostaa omaa tekemistään eikä anna ulkopuolisten väheksyä heidän roolejaan ja toimintaansa. Kuten fasilitaattorin, myös opettajan roolia tulee hoitaa kevyesti, tiimin itseohjautumista kunnioittaen. (Adkins 2010, 145, 147, 180–181.)

Scrumiin ei kuulu se, että Scrum-mestari ratkaisisi ongelmia tiimin puolesta niiden ilmaannuttua saati etukäteen. Hänen tulee hoitaa *ongelmanratkaisijan* roolia tuomalla esiinnousseet ongelmat kehitystiimin tietoisuuteen. Varsinainen ongelmien ratkaiseminen tai ratkaisematta jättäminen on tiimin velvollisuus. Mikäli Scrum-mestari huomaa ongelman tai saa kuulla mahdollisesta ongelmasta, hän voi asiaa harkittuaan kertoa siitä tiimille, jonka tehtävänä on arvioida kannattaako asiasta huolestua. Scrum-mestarin pitää luottaa tiimiin ja siihen, että he tietävät käytännön

asioista paremmin kuin hän itse. Toki tässä täytyy ottaa huomioon tiimin kokemus. Tarvittaessa kokematonta tiimiä voidaan opettaa todellisten ongelmien kautta. (Adkins 2010, 183, 185–186.) Scrum-mestarin täytyy muistaa, että kehitystiimin tehokkuutta häiritsevien haittatekijöiden ratkaiseminen kuuluu hänen tärkeimpiin tehtäviinsä eikä niiden ratkaiseminen kuulu tiimin vastuulle. Mikäli tiimin jäsenet kertovat jostain ongelmasta, johon Scrum-mestari voi vaikuttaa, hänen pitää puuttua siihen pikimmiten. Haittatekijöitä voivat olla esimerkiksi kehitystiimin käyttämiin tietokoneisiin ja tietoverkkoihin liittyvät ongelmat, Scrum-tiimin ulkopuolelta tulevat vaatimukset sprintin tuotokseen liittyen tai epävarmuus etenemisestä, käytetyistä teknologioista tai suunnitelmista. (Schwaber & Beedle 2001, 44–45.)

Jokaisessa tiimissä nousee ajoittain esille konflikteja. Erilaiset ristiriidat ovat työelämään kuuluvia arkipäiväisiä ja normaaleja tilanteita, kunhan ne eivät huononna työtovereiden keskinäisiä ihmissuhteita. Konflikteja ei siis pidä kavahtaa, sillä ne ovat organisaation elinehto, osa aktiivista, aikaansaavaa ja kehittyvää työelämää. (Parvikko 2001, hakupäivä 29.3.2011.) Kuten kaikissa muissakin tiimeissä myös Scrum-tiimeissä tulee aika ajoin erilaisia tiimin jäsenten välisiä ristiriitatilanteita. Tämä on ymmärrettävää siitäkin syystä, että tiimin jäsenillä on erilaiset taustat, he ovat oman alansa asiantuntijoita, joilla voi olla erilaisia mielipiteitä, ja he ovat päivittäin tiiviisti tekemisissä toistensa kanssa. Scrum-mestarin tulee *konfliktien selvittäjänä* auttaa tiimiä selviytymään tilanteesta. Hänen ei kuitenkaan tarvitse puuttua konfliktiin välittömästi, vaan hän voi seurata, saavatko asianosaiset ratkaistua ristiriitatilanteen itsenäisesti. Mikäli tämä ei onnistu, hän voi ohjata ratkaisua auttamalla konfliktin osapuolia näkemään tilanteen ja auttamalla heitä selvittämään sen. Hänen on myös tärkeä kertoa tiimiläisille erilaisista konfliktinratkaisutavoista, jotta nämä osaavat itse jatkossa toimia rakentavammin. (Adkins 2010, 203–204, 226.)

Scrum-mestarin tärkeänä roolina on toimia *yhteistyön johdattelijana*. Yhteistyön ajatuksena on, että yhdessä toimimalla saadaan aikaiseksi enemmän kuin yksittäisten osien summa. Tiimin jäsenten tulisi ideoida aidosti yhdessä siten, että jokainen tuo esille oman näkemyksensä asioiden parantamiseksi. Scrum-mestari voi toiminnallaan määritellä sävyn, jolla erilaisia näkemyksiä käsitellään. Hän voi myös opastaa tiimiläisiä yhteistyötaitojen kehittämisessä, kannustaa heitä arvostamaan itseään ja osaamistaan, rohkaista heitä tuomaan mielipiteitään esille ja auttaa heitä tilanteissa, joissa yhteistyö on jumiutunut. (Adkins 2010, 229, 231–232, 253.)

3.3.2 Tuoteomistaja

Tuoteomistaja (Product Owner) on tuotteen kehitykseen osallistuva, sidosryhmien valtuuttama henkilö. Mikäli kyseessä on asiakasprojekti, tuoteomistajan tulisi olla asiakkaan edustaja. (Virtanen 17.6.2010, luento.) Joissakin projekteissa tuotteen asiakkaina voi olla jopa miljoonia ihmisiä, joilla on moninaisia tarpeita. Tällaisissa projekteissa tuoteomistajan rooli on hyvin samankaltainen kuin tuotepäällikön tai tuotteen markkinointijohtajan. (Deemer ym. 2010, 6.) Oleellista on, että roolia hoitaa aina yksi ihminen, ei tiimi tai toimikunta. Tietyissä tapauksissa tuoteomistaja voi olla yksi kehitystiimin jäsenistä, mutta hän ei saa koskaan toimia myös Scrum-mestarina. (Schwaber & Sutherland 2010, 7.)

Tuoteomistajan tehtävät ovat vaativia, mutta palkitsevia. Hän on luvun 3.3 tarinan mukainen "sika", joka on vastuussa tuotteen työlistasta ja sen tärkeysjärjestyksestä. (Schwaber & Sutherland 2010, 5.) Hän kerää tuotteeseen liittyviä ideoita sidosryhmiltä ja päättää, mitkä niistä otetaan tuotteen työlistalle (Virtanen 17.6.2010, luento). Kukaan muu ei voi tehdä muutoksia tuotteen työlistaan, vaikka muut voivatkin yrittää vaikuttaa tuoteomistajaan. Jotta tuoteomistaja voi toteuttaa tämän tehtävänsä hyvin, tulee kaikkien organisaation jäsenten kunnioittaa hänen päätöksiään. (Schwaber & Sutherland 2010, 7.) Tuoteomistajalla tulee olla käytännön projekti-työkokemuksen lisäksi taloudelliseen päätöksentekoon liittyvää kokemusta. Tätä tarvitaan tuotteen työlistan ominaisuuksien tärkeysjärjestystä määriteltäessä. (Vrt. tuotteen työlistasta kertova luku 3.5.1.)

Tuoteomistajan tehtävänä on myös huolehtia, että jokainen kehitystiimin jäsen tuntee tuotteen työlistan ominaisuuksien tärkeysjärjestyksen, ja mitä niistä toteutetaan milloinkin. Kehitystiimi ei saa kuunnella muiden mielipiteitä tässä asiassa. (Schwaber & Sutherland 2010, 7.)

3.3.3 Kehitystiimi

Kaikki tuotteen toteuttamiseen liittyvä työ tehdään kehitystiimissä, joka on sprintin aikana tapahtuvasta työstä vastaava "sika" (Schwaber & Sutherland 2010, 5, 8). Tiimiä perustettaessa Scrum-mestarin on huolehdittava siitä, että tiimi tietää toimintavaltuutensa: mitä tiimi voi ja vastaavasti ei voi tehdä ja päättää itse (Lämsä & Hautala 2008, 129).

Kehitystiimi on täysin itsehallinnollinen, oma-aloitteinen ja itseohjautuva (Schwaber & Beedle 2001, 9). Itseohjautuvuus tarkoittaa sitä, että tiimillä on laajat toimintavaltuudet. Tällöin tiimiläiset päättävät itsenäisesti päivittäisestä toiminnasta. (Lämsä & Hautala 2008, 129.) Jotta nämä ominaisuudet toteutuvat, tulee tiimin jäsenillä olla ammattiosaamista eri osa-alueilta. Kehitystiimin pitää selvitä monimutkaisista vaatimuksista ja edistyneiden teknologioiden käytöstä. Voidakseen toimia tehokkaasti, tiimin jäsenten täytyy oppia tuntemaan toisensa ja luottamaan itseensä. Kehitystiimiläisten pitää toimia rehellisesti ja päättäväisesti, kompromisseja ja epäonnistumisia ei saa pelätä. (Schwaber & Beedle 2001, 7, 9, 147.)

Scrumissa kehitystiimin jäsenten on selvitettävä itsenäisesti, miten he voivat olla mahdollisimman tuottavia. Suunnittelu- ja toteutustyö kuuluu täysin tiimin vastuulle. Scrum-mestari ja muut voivat opastaa ja neuvoa tiimiä, mutta tiimin tulee itse johtaa itseään ja päättää, miten he haluavat työnsä tehdä. (Schwaber 2004, 101.) Kenelläkään kehitystiimin ulkopuolisella henkilöllä – ei edes Scrum-mestarilla – ole oikeutta määritellä miten työ tulee käytännössä tehdä. Tämä on tiimin itsensä selvitettävä. (Schwaber & Sutherland 2010, 8.)

Kehitystiimin toimintaa rajoittavat vain organisaatiossa käytössä olevat normit ja käytännöt sekä tiimin itsensä tuotteen työlistalta valitsemat tehtävät (Schwaber & Beedle 2001, 9). Yhden sprintin aikana kehittäjien muodostama tiimi työstää valitsemansa tuotteen työlistan ominaisuudet julkaisuvalmiiksi palaseksi valmista tuotetta (Schwaber & Sutherland 2010, 8).

Kehitystiimin jäsenillä ei saa olla titteleitä tai arvoasemia, vaan heidän kaikkien tulee olla saman-arvoisessa asemassa toisiinsa nähden. Kaikkein tärkein tiimiä yhdistävä taito on valmius käsitellä vaatimuksia ja tuottaa niiden pohjalta käyttökelpoinen tuote. Ammattiosaamiseltaan kehitystiimin jäsenet ovat usein erikoistuneet eri osa-alueisiin kuten ohjelmointiin, testaukseen, liiketalouteen, järjestelmäarkkitehtuuriin, käyttöliittymäsuunnitteluun tai tietokantasuunnitteluun. Tiimin sisälle ei kuitenkaan saa muodostua osatiimejä, jotka toteuttavat jotakin tiettyä aihetta kuten testausta. Scrumin periaatteena on, että kukin kehitystiimin jäsen tuo oman ammattitaitonsa ongelma-tilanteiden ratkaisuun. Tällöin yhteisvaikutuksena on koko tiimin tehokkuuden paraneminen. Tiimin toiminnan kannalta onkin tärkeää, että jäsenet eivät pitäydy liikaa omissa kiinnostuksen kohteissaan, vaan ovat valmiita opettelemaan uusia asioita. (Schwaber & Sutherland 2010, 8.) Näin tiimin toiminta ei ole riippuvaista tietystä jäsenestä, vaan esimerkiksi irtisanoutumis- tai sairaustapauksissa muut tiimin jäsenet voivat tarvittaessa tehdä kyseisen työntekijän vastuulla olleita tehtäviä.

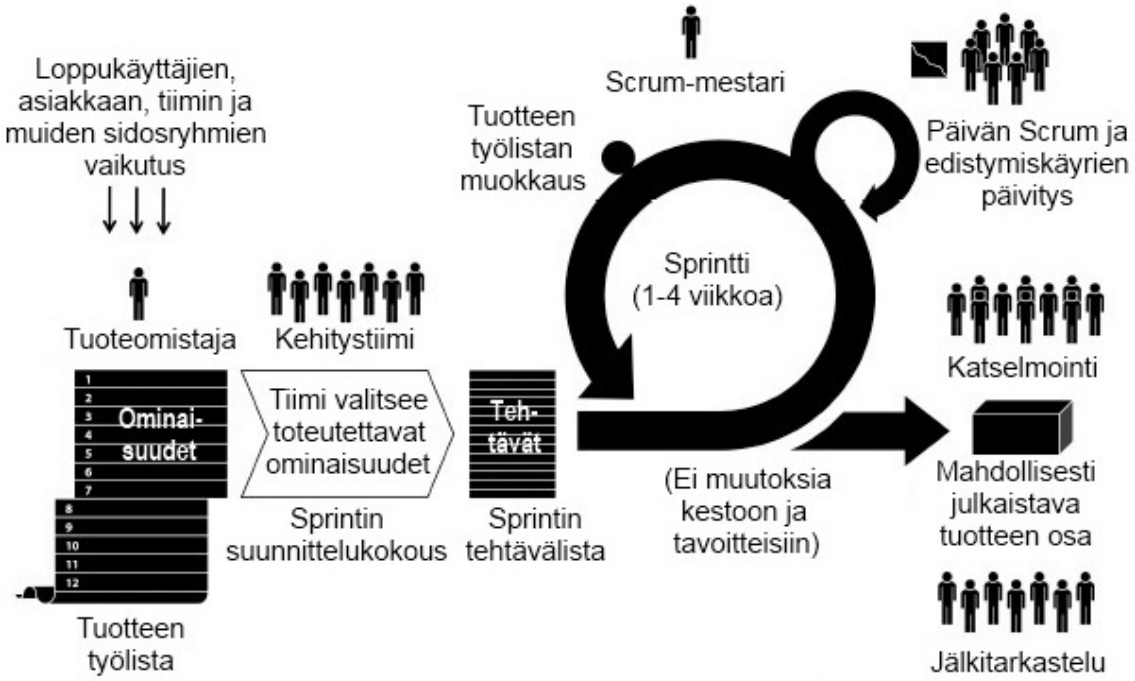
Schwaberin ja Sutherlandin mukaan ihanteelliseen kehitystiimiin kuuluu seitsemän jäsentä, mutta myös viiden–yhdeksän jäsenen tiimit ovat toimivia. Tähän lukuun ei lasketa Scrum-mestaria ja tuoteomistajaa, mikäli he eivät osallistu tiimin toimintaan myös jäsenen roolissa. Jos kehitystiimissä on vähemmän kuin viisi jäsentä, ei tiimissä ole samanlaista vuorovaikutusta kuin hieman suuremmassa tiimissä. Tällöin toiminta ei ole yhtä tehokasta kuin ihanteellisen kokoisessa tiimissä. Liian pienessä tiimissä vaarana on myös tilanne, jossa tiimin jäsenillä ei olekaan tarvittavaa tietotaitoa tuotteen jonkin osa-alueen toteuttamiseen. Mikäli jäseniä on enemmän kuin yhdeksän, haittana on yhteistyön hankaloituminen. Suuret kehitystiimit voivat olla liian monimutkaisia kokemusperäisen prosessin onnistumiseen. (2010, 8.)

Oli tiimin koko mikä hyvänsä, jäsenet voivat vaihtua sprinttien välillä. Tällöin pitää ottaa huomioon, että kun tiimin jäsenet muuttuvat, itseohjautumisella saavutettu tuottavuus heikkenee joksikin aikaa. Tämän vuoksi pitääkin harkita tarkkaan, onko tiimin kokoonpanon muuttaminen tarpeellista ja voitaisiinko siltä välttyä. (Schwaber & Sutherland 2010, 9.)

Toisiin tiimiläisiin ja heidän vahvuuksiinsa tutustumiseen kannattaa panostaa. Kun tiimiläiset tuntevat toisensa ja luottavat toisiinsa, saadaan itseohjautumistakin helpotettua. Virtanen (17.6.2010, luento) ohjeistaa, että mikäli kaikki sprintin toteuttamiseen osallistuvan kehitystiimin jäsenet eivät entuudestaan tunne toisiaan, voidaan järjestää niin sanottu kick off -päivä, jonka aikana tiimiläiset tutustuvat toisiinsa ja sopivat yhteisistä säännöistä.

3.4 Scrumin prosessikuvaus

Scrumin periaatteiden mukaisesti etenevä prosessi on kuvattu kuviossa 8. Kuvio on alkuperäisen lähteen mukainen, mutta termit on käännetty suomeksi. Kuviossa esitetään tiiviissä muodossa Scrum-tiimin roolit, projektin aikaiset tapahtumat ja prosessiin kuuluvat dokumentit. Kuviossa ei ole esitelty julkaisun suunnittelukokousta, jonka ei hyödyllisyydestään huolimatta katsota olevan välttämätön.



KUVIO 8. Scrum-prosessi (Deemer, Benefield, Larman & Vodde 2010, 5, termit suomennettu)

Scrum-projekti, kuten muutkin projektit, lähtee liikkeelle suunnittelusta. Schwaberin mukaan vähimmäisvaatimus projektin aloitukselle on visio toteutettavasta tuotteesta ja tuotteen työlistan laatiminen. Vision avulla saadaan käsitys, miksi projektiin ryhdytään ja mikä on sen toivottu lopputulos. Visio kuvaa esimerkiksi sitä, miten asiakkaan liiketoiminta ja asema markkinoilla paranee uuden tuotteen käyttöönoton myötä tai mitkä ovat tuotteen pääominaisuudet ja -toiminnot ja miten ne hyödyttävät asiakasta. Tuotteen työlista määrittelee ne toiminnalliset ja ei-toiminnalliset vaatimukset, joiden avulla visio voidaan toteuttaa. (2004, 68.) Tuotteen työlistaan palataan tarkemmin luvussa 3.5.1.

Yksi suunnitelman tavoitteista on saada asiakas tai rahoittaja sijoittamaan projektiin. Suunnitelman tulee vakuuttaa asiakas tai rahoittaja siitä, että projektilla on arvoa, tiettyjä asioita toimitetaan tiettyyn aikaan, hyödyt ovat kuluja ja riskejä suuremmat ja projektiin osallistuva henkilöstö on kyllin pätevää suunnitelman toteuttamiseen. (Schwaber 2004, 68.)

Scrum voidaan ottaa käyttöön myös siinä vaiheessa, kun kyseessä oleva projekti on jo visioitu ja esitelty asiakkaalle tai rahoittajalle. Tällöin rahoitus on jo kunnossa ja odotukset selvillä. Tämän jälkeen on välttämätöntä suunnitella projektia Scrumin mukaisesti, jotta kehitystiimi, tuoteomistaja ja sidosryhmien jäsenet voivat nähdä projektin sarjana kehitysjaksoja, jotka johtavat tuotteen julkaisuun. (Schwaber 2004, 68–69.)

3.4.1 Julkaisun suunnittelukokous

Julkaisun suunnittelukokouksessa (Release Planning Meeting) määritellään toteutettavan julkaisun eli lopputuotteen suunnitelma ja tavoitteet niin selkeästi, että Scrum-tiimi ja muu organisaatio ymmärtävät ne. Julkaisua suunniteltaessa mietitään, kuinka visio voidaan muuttaa voitokkaaksi tuotteeksi sekä kuinka voidaan saavuttaa tai ylittää toivottu asiakastyytyväisyys ja sijoitetun pääoman tuotto. Julkaisusuunnitelmassa määritellään tavoite, tuotteen työlistan tärkeysjärjestys, suurimmat riskit, julkaisun ominaisuudet ja toiminnallisuudet, todennäköinen toimituspäivä ja kustannusarvio. Suunniteltuja ominaisuuksia ja toiminnallisuuksia ei kuitenkaan lyödä lukkoon, vaan suunnittelua jatketaan sprintin katselmoinnissa ja suunnittelukokouksessa sekä jokaisessa Päivän Scrumissa. (Schwaber & Sutherland 2010, 9–10.) Huolellinen, projektin edetessä mukautuva suunnittelu parantaa julkaisun onnistumista. Tällöin projekti etenee joustavasti ja asiakkaan toiveet sekä eteen tulevat ongelmat huomioidaan.

Julkaisun suunnittelukokous ei ole välttämätön, vaikkakin erittäin hyödyllinen. Mikäli suunnittelukokousta ei pidetä, Scrum-tiimi tulee kohtaamaan erilaisia suunnittelemattomuudesta johtuvia haittoja, jotka pitää käsitellä ennen kuin projekti voi jatkua. Näin ollen haittojen selvittäminen on yksi tuotteen työlistan luomisen vaiheista. (Schwaber & Sutherland 2010, 9.)

3.4.2 Sprintin suunnittelukokous

Jokaisen sprintin aluksi pidetään sprintin suunnittelukokous (Sprint Planning Meeting), jossa suunnitellaan alkavan sprintin lopputulos ja toteutus. Kokoukseen osallistuvat tuoteomistaja ja kehitystiimi, mutta lisäksi mukaan voidaan ottaa myös Scrum-tiimin ulkopuolisia asiantuntijoita, mikäli tiimi haluaa pyytää heiltä neuvoa. (Schwaber & Sutherland 2010, 11, 13.)

Schwaberin ja Sutherlandin mukaan suunnittelukokouksen enimmäiskestoaika määräytyy sen mukaan, minkä mittaisesta sprintistä on kyse. Esimerkiksi kuukauden mittaisen sprintin suunnittelukokoukselle varataan aikaa kahdeksan tuntia. Ensimmäisen neljän tunnin aikana kehitystiimi ja tuoteomistaja määrittelevät, *mikä* on sprintin haluttu lopputulos. (2010, 11–12.) Lopputuloksen määrittely lähtee liikkeelle tuoteomistajan esitellessä tuotteen työlistan tärkeimmät ominaisuudet kehitystiimille (Deemer ym. 2010, 9). Jälkimmäisellä puoliskolla kehitystiimiläiset miettivät, *miten* lopputulokseksi valittu tuotteen osa saadaan toteutettua (Schwaber & Sutherland 2010, 12). He keskustelevat ominaisuuksien tarkoituksesta, millä kriteereillä ne voidaan katsoa valmiiksi ja siitä,

miksi tuoteomistaja toivoo niiden toteuttamista seuraavan sprintin aikana. (Deemer ym. 2010, 9). Tuoteomistajan ei välttämättä tarvitse olla paikalla suunnittelukokouksen toisessa osassa, mutta tarvittaessa häneltä on voitava kysyä selvennyksiä esimerkiksi puhelimitse (Schwaber & Sutherland 2010, 12).

Jotta tiedetään, kuinka monta työtuntia sprintin aikana voidaan työskennellä, palaverin jälkimmäisen osan aluksi käydään läpi ketkä sprinttiin osallistuvat, tiimin aikaisempi suorituskyky ja jo toteutetut tuotteen osat (Schwaber & Sutherland 2010, 12). Lisäksi jokaiselta tiimin jäseneltä kysytään, montako tuntia hän voi tehdä töitä sprintin aikana. Lomien ja muiden tiedossa olevien vapaapäivien lisäksi todellista työaikaa vähentävät myös ruokatauot, palaverit ja muut menot. (Virtanen 17.6.2010, luento).

Tämän jälkeen toteutetaan yksi Scrumin tärkeimmistä käytännöistä: tiimi päättää itsenäisesti millaisen määrän työtä se ottaa vastuulleen sprintin ajaksi. Tuoteomistajalla ei ole sananvaltaa tässä asiassa, mutta hänen tulee kuitenkin pitää huolta, että tiimi valitsee tuotteen työlistalta korkeimmilla prioriteeteilla olevia ominaisuuksia. Mikäli kehitystiimi huomaa, että jokin ominaisuus tulisi esimerkiksi teknisistä syistä toteuttaa ehdottomasti ennen jotain muuta ominaisuutta, voidaan asiasta keskustella tuoteomistajan kanssa. (Deemer ym. 2010, 9.)

Projekti tulee aloittaa toteuttamalla ensimmäisenä nopeimmin lisäarvoa tuottava ja riskialtein julkaisun osa (Schwaber & Sutherland 2010, 9). Näin ollen jo ensimmäisen kehitysjakson aikana toteutetaan uutta toiminnallisuutta tuotteeseen. Järjestelmäarkkitehtuuri ja ulkoasu kehittyvät usean sprintin aikana, joten yksistään niihin ei ensimmäisessä sprintissä pidä keskittyä. (Schwaber & Beedle 2001, 7–8.)

Sprintin suunnittelukokouksen lopputuotoksena on sprintin tehtävälista (katso luku 3.5.3), jossa on listattuna kaikki sprintin aikana toteutettaviksi otetut ominaisuudet aika-arvioineen. Tuotteen työlistan työstäminen aloitetaan jakamalla korkeimman prioriteetin ominaisuus itsenäisiin tehtäviin. (Deemer ym. 2010, 9–10.) Jokaisen tehtävän tulee olla niin pieni, että se voidaan toteuttaa enintään yhden työpäivän aikana (Schwaber & Sutherland 2010, 13). Tiimin tuleekin arvioida, kuinka monta tuntia kunkin tehtävän tekemiseen kuluu. Ominaisuuksia valitaan ja pilkotaan niin monta, kuin tiimi arvioi voivansa toteuttaa sprintille lasketussa työtuntimäärässä. (Deemer ym. 2010, 10.)

3.4.3 Sprintti

Scrumin käytäntöjen mukaisesti projektia viedään eteenpäin enintään 30 päivän mittaisten sprinttien (Sprint) avulla (Schwaber & Beedle 2001, 7). Sprinttien kesto pidetään lyhyenä, jotta projektin rahoittajat ja kehitykseen osallistuvat tuotteen käyttäjät voivat nähdä uudet toiminnallisuudet riittävän nopeasti. Tällöin he eivät menetä mielenkiintoaan ja sitoutuvat projektiin. (Schwaber 2004, 55.)

Jokaisen sprintin eli kehitysjakson tuloksena tulee olla uusi, käyttökelpoinen tuotteen toiminnallisuus tai osa (Schwaber & Beedle 2001, 8). Lopputuloksen toteuttamisen kaikkien osa-alueiden – vaatimusten, suunnittelutyön, ohjelmoinnin, testauksen ja dokumentaation – pitää olla tehtyinä etukäteen sovittujen kriteerien mukaisesti (Schwaber 2004, 55).

Tiimin tulee työskennellä sprintin kehitysvaiheen ajan suunnittelukokouksessa asetettujen tavoitteiden mukaisesti. Tiimi on sitoutunut tekemään töitä keskeytyksettä. (Schwaber 2004, 101.) Koska työlistalta valittujen ominaisuuksien määrä perustuu tiimin omaan analysointiin ja päätökseen, tiimin jäsenet sitoutuvat vahvemmin töiden valmistumiseen (Deemer ym. 2010, 9). Mikäli työ kuitenkin osoittautuu haastavammaksi kuin mitä ennakkoon ajateltiin, voi kehitystiimi yhdessä tuoteomistajan päättää kanssa, miten tavoitetta muutetaan (Schwaber & Sutherland 2010, 12).

Kehitystiimi organisoi itse kuka tekee mitäkin sprintin tehtävälistan tehtävää. Tämä voidaan tehdä joko sprintin suunnittelukokouksessa tai myöhemmin sprintin edetessä. Scrum-mestarin tehtävänä on varmistaa, että suunnitelmiin ei tehdä sprintin lopputulokseen vaikuttavia muutoksia. Myös sekä tiimin kokoonpanon että laatuvaatimusten tulee pysyä muuttumattomina koko sprintin ajan. (Schwaber & Sutherland 2010, 10, 13.) Toisinaan tämä ei kuitenkaan ole mahdollista, sillä joku työntekijöistä voi esimerkiksi sairastua tai siirtyä toisen yrityksen palvelukseen.

Joskus sprintin laajuutta päätetään mukauttaa Scrum-tiimin yhteisellä päätöksellä. Voidaan jopa päätyä tilanteeseen, jossa sprintti joudutaan lopettamaan kesken kaiken. Mikäli tilanne todella vaatii keskeyttämistä, tulee siihen olla riittävästi rohkeutta. Yksi mukauttamiseen tai keskeyttämiseen johtavista syistä voi olla tiimin näkökanta, että se ei voi saavuttaa sprintin tavoitteita. Joskus myös ulkopuoliset olosuhteet tekevät sprintin tavoitteesta epäoleellisen. Näin voi käydä esimerkiksi jos kilpailutilanteessa, liiketoiminnassa tai teknologioissa tapahtuu muutoksia. Vaikka koko tiimi on mukana keskeyttämispäätöksessä, niin lopullisen päätöksen voi tehdä vain tuote-

omistaja. Mikäli sprintin keskeyttämiseen päädytään, tulee järjestää uusi suunnittelupalaveri, jossa sprintin tavoitteet muodostetaan uudelleen. (Tieturi Oy 2010, 84.)

3.4.4 Päivän Scrum

Päivän Scrum -palaveri (Daily Scrum Meeting) on erinomainen paikka arvioida sprintin edistymistä (Schwaber & Beedle 2001, 8). Lisäksi sen tarkoituksena on parantaa kommunikaatiota, karsia tehokasta työaikaa vieviä tapaamisia, tunnistaa ja poistaa työskentelyn esteitä, tehostaa ja edistää nopeaa päätöksentekoa ja kohentaa kaikkien tiimin jäsenten tietämystä projektin etenemisestä (Schwaber & Sutherland 2010, 15).

Päivän Scrumin periaatteena on, että Scrum-mestari ja kaikki kehitystiimin jäsenet osallistuvat joka päivä samaan kellonaikaan järjestettävään, enintään 15 minuutin mittaiseen palaveriin. Jotta palaverin kesto pysyy annetuissa rajoissa, on suositeltavaa, että kaikki osallistujat seisovat. (Deemer ym. 2010, 12.) Tämän vuoksi Päivän Scrumia kutsutaan joissakin lähteissä "stand-upiksi" (Adkins 2010, 119).

Kehitystiimin on huolehdittava palaverin kulusta itsenäisesti. Jokainen kehitystiimin jäsen vastaa vuorollaan lyhyesti kolmeen kysymykseen:

1. Mitä olen saanut tehtyä edellisen päiväpalaverin jälkeen?
2. Mitä aion tehdä ennen seuraavaa päiväpalaveria?
3. Mitä esteitä olen kohdannut?

Näiden kysymysten lisäksi palaverissa ei keskustella muista asioista. (Deemer ym. 2010, 12.) Mikäli jokin palaverissa esiin nousseista asioista edellyttää muutosten tekemistä sprintin tuleviin tehtäviin, voidaan asiasta keskustella lisää erillisessä palaverissa (Schwaber & Sutherland 2010, 15).

Kehitystiimin jäsenten lisäksi myös Scrum-mestari saa Päivän Scrumissa käsityksen prosessin kulusta ja mahdollisista etenemisesteistä. Näin hän voi opastaa tiimin jäseniä hyvissä ajoin, jotta sprintin lopputulos ei vaarannu. (Leffingwell & Smits 2005, 4.) Lisäksi Scrum-mestarin tehtävänä on varmistaa, että kehitystiimi tuntee palaverin säännöt ja että tiimi pitää palaverin päivittäin (Schwaber & Sutherland 2010, 15).

Deemerin, Benefieldin, Larmanin ja Vodden kanta on, että vain Scrum-tiimin jäsenet osallistuvat Päivän Scrumiin. Mikäli paikalla on johtajia tai muita arvovaltaisia henkilöitä, kehitystiimin jäsenet voivat kokea olevansa tarkkailun alla ja tuntea painetta kertoa vain työn hyvään edistymiseen liittyvistä asioista. Tällöin Päivän Scrumin keskeinen tehtävä avoimuutta ja itseohjautumista parantavana palaverina voi jäädä toteutumatta. (2010, 12.) Schwaberin ja Sutherlandin mielestä palaveriin voi osallistua myös muita sidosryhmien edustajia. Tällöin Scrum-mestarin tulee kuitenkin pitää huolta, että "kanat" eivät sekaannu palaverin kulkuun esimerkiksi ottamalla kantaa asioihin. (2010, 15.)

3.4.5 Sprintin katselmointi

Sprintin loppuvaiheessa järjestetään sprintin katselmointi (Sprint Review Meeting). Tämä luonteeltaan epämuodollinen kokous antaa arvokasta pohjatietoa seuraavan sprintin suunnitteluun. (Schwaber & Sutherland 2010, 14.) Katselmoinnissa esitellään lyhyesti kehitystiimin aikaansaama tuotteen osa. Palaverin pääpaino on kuitenkin avoimessa keskustelussa eikä tuotteen demoamisessa. (Deemer ym. 2010, 14.)

Katselmoinnin aluksi tuoteomistaja esittelee sprintin tehtävälistan avulla, mitä sprintin tehtävistä on saatu tehtyä ja mitä jäi tekemättä. Tämän jälkeen Scrum-tiimi keskustelee siitä, mikä sujui hyvin, mitä ongelmia kohdattiin ja miten ongelmat ratkaistiin. (Schwaber & Sutherland 2010, 13–14.) Katselmoinnin tärkeintä antia on kehitystiimin ja tuoteomistajan välinen syvällinen keskustelu tuotteen työlistan ja projektin tilanteesta (Deemer ym. 2010, 14). Edistymiskäyrien perusteella arvioidaan tulevien sprinttien kehitysnopeutta ja valmiin tuotteen julkaisuaikaa (Schwaber & Sutherland 2010, 14).

Palaverin ajankohdasta ja paikasta sovitaan hyvissä ajoin. Keskustelun johdattelijana toimii Scrum-mestari. Katselmoinnissa esille tulleet asiat kirjataan ylös esimerkiksi teknisen kirjoittajan tai dokumentoijan toimesta. Näin katselmoinnin kulusta saadaan raportti myöhempää tarkoitusta ja mahdollisesti poissaolevia sidosryhmien jäseniä varten. (Virtanen 17.6.2010, luento.)

Katselmoinnin ohjeellinen kesto on neljä tuntia, kun kyseessä on kuukauden mittainen sprintti. Vastaavasti esimerkiksi kahden viikon mittaisen sprintin katselmointikokous kestää kaksi tuntia. (Schwaber & Sutherland 2010, 13.)

3.4.6 Sprintin jälkitarkastelu

Sprintin jälkitarkastelussa (Sprint Retrospective Meeting) kehitystiimi ja Scrum-mestari tutkivat, miten Scrum-prosessi toimi edellisen sprintin aikana, ja pyrkivät löytämään parantamiskohteita seuraavaa sprinttiä varten (Schwaber 2004, 102). Näin seuraavista sprinteistä yritetään tehdä edellisiä tehokkaampia ja miellyttävämpiä. Jälkitarkastelun tavoitteena on saada aikaan avointa keskustelua Scrum-tiimin kokoonpanosta, henkilösuhteista, kommunikaatiosta, käytetyistä työkaluista ja siitä, miten sprintin tehtävälistan tehtävät saatiin tehtyä valmiiksi. (Schwaber & Sutherland 2010, 14.)

Tuoteomistajan läsnäolo ei ole välttämätöntä, mutta hänkin on halutessaan tervetullut palaveriin. Scrum-mestarin tehtävänä on johdatella keskustelua kantaa ottamatta. Mikäli organisaatiossa on useita Scrum-mestareita, he voivat osallistua ristiin toistensa jälkitarkastelupalavereihin. Näin keskustelun johdattelu on neutraalimpaa. (Deemer ym. 2010, 14.)

Virtasen mukaan Scrumissa ei oteta kantaa siihen, miten jälkitarkastelu tulisi toteuttaa. Tapaamisessa voidaan esimerkiksi hyödyntää aivoriihitekniikkaa, jolloin ujoimmatkin tiimin jäsenet saavat sanottua ajatuksensa, mutta tekniikkaa voidaan myös vaihdella, jolloin tiimin mielenkiinto säilyy yllä sprintistä toiseen. (17.6.2010, luento.) Schwaber ja Sutherland (2010, 14) ehdottavat, että osallistujat voivat nostaa esille ja asettaa tärkeysjärjestykseen asiat, jotka menivät hyvin, ja asiat, joita parantamalla työskentely sujuisi entistä paremmin. Deemer ym. opastavat, että tämä voidaan tehdä esimerkiksi piirtämällä piirtotaululle kaksi saraketta, joista toinen otsikoidaan "Mikä sujui hyvin?" ja toinen "Mikä voisi mennä paremmin?". Tämän jälkeen kukin osallistuja lisää yhden tai useamman asiaan kumpaankin listaan. Mikäli sama asia otetaan esille useaan kertaan, merkitään kyseiseen kohtaan ruksi. Näin asiat, joista monet ovat samaa mieltä, näkyvät konkreettisesti. Lopuksi tiimiläiset sopivat yhdessä muutamista asioista, joihin pyritään tekemään muutos tulevan sprintin aikana. Valitut asiat käsitellään uudestaan seuraavassa jälkitarkastelupalaverissa, ja katsotaan onko tuloksia saavutettu. (2010, 14.)

Jälkitarkasteluun käytettävä ohjeellinen aika on kolme tuntia, kun sprintin kesto on kuukausi (Schwaber & Sutherland 2010, 14). Aikaraja on määritelty sen vuoksi, ettei Scrum-tiimi sorru käyttämään liikaa aikaa toimintansa hiomiseen. Täydellisyyttä ei nimittäin voida koskaan saavuttaa, joten on hyväksyttävä se tulos, joka saadaan aikaiseksi sovitun ajan puitteissa. (Schwaber 2004, 110.)

3.5 Scrumin dokumentit

Scrumin käytäntöihin kuuluu se, että suunnitelmista ei tehdä laajoja paperidokumentteja. Suunnittelua voidaan tehdä vain nopeasti esimerkiksi isoille papereille tai tussitaululle. Ketteriä menetelmiä hyödynnettäessä on ymmärrettävä, että suunnitelmat kehittyvät ja muuttuvat, joten laajamittainen dokumentointi ei ole tarpeen. Tämä ei kuitenkaan tarkoita sitä, että valmiin tuotteen dokumentaatio voidaan jättää tekemättä. (Virtanen 17.6.2010, luento.)

Tuotteen lopullisen dokumentaation lisäksi Scrumiin liittyy menetelmää varten kehitettyjä dokumentteja, joilla parannetaan projektin avoimuutta ja läpinäkyvyyttä. Nämä dokumentit ovat tuotteen työlista, julkaisun edistymiskäyrä, sprintin tehtävälista ja sprintin edistymiskäyrä. Niiden lisäksi projektin tiedoista viestitään myös kasvokkain esimerkiksi Päivän Scrumeissa ja sprintin katselmointikokouksissa. (Schwaber 2004, 83.)

Työ- ja tehtävälistojen sekä edistymiskäyrien lisäksi jokaisen sprintin lopuksi luodaan muodolliset raportit. Nämä raportit tarjoavat tilannekatsauksen projektin kulkuun. Niiden tarkoituksena on pitää jokainen projektista kiinnostunut sidosryhmän jäsen ajan tasalla. (Schwaber 2004, 83.)

Scrum ei määrittele listoille yhtä tiettyä, oikeaa ulkoasua. Scrum-mestarin tehtävänä on pitää tarvittavat tiedot näkyvillä. Hän laatii listapohjan, jota muu organisaatio alkaa käyttää. Hän voi käyttää siihen organisaatiossa jo entuudestaan tuttua kieltä ja työkalua, kuten Microsoft Exceliä tai Microsoft Projectia. Tällöin projektiin osallistujien ei tarvitse opetella täysin uutta raportointityökalua ja työntekijät oppivat Scrumin raportointikäytännöt helpommin. (Schwaber 2004, 91, 99.)

3.5.1 Tuotteen työlista

Tuotteen työlistassa (Product Backlog) luetteloidaan ominaisuudet ja toiminnot, jotka valmiin tuotteen tulisi sisältää (Schwaber & Beedle 2001, 7). Siinä käsitellään myös tekniset ja ei-toiminnalliset vaatimukset. Tekniset vaatimukset voivat liittyä esimerkiksi ohjelmisto-arkkitehtuuriin. Ei-toiminnallisia ominaisuuksia ovat kaikki kehitystiimille työtä aiheuttavat laatuvaatimukset, kuten ohjelmiston käytettävyyteen, luottamuksellisuuteen, eheyteen, ulkoasuun ja suorituskykyyn liittyvät asiat. (Tieturi Oy 2010, 39.) Lisäksi työlistaan kirjataan projektin edetessä huomatut ohjelmointi- ja muut virheet tarvittavine korjauksineen. Tuotteen työlista siis kehittyy

jatkuvasti tuotekehityksen mukana ja on olemassa niin kauan kuin tuotekin. (Schwaber & Sutherland 2010, 16.)

Tuotteen työlistan toiminnallisten ominaisuuksien miettimisessä voidaan hyödyntää käyttäjätarinoita (User Stories). Ne ovat julkaisun suunnittelukokouksessa yksinkertaisilla virkkeillä paperille kirjattuja vaatimuksia, joissa kerrotaan käyttäjän rooli, mitä hän haluaa tehdä ja miksi. Toiminnallisen vaatimuksen kuvaava käyttäjätarina voi kuulua esimerkiksi: "Asiakkaana haluan nähdä tarjolla olevat koulutukset, jotta voin selvittää mitkä niistä ovat minulle hyödyllisiä." Esimerkki ei-toiminnallisesta käyttäjätarinasta on: "Asiakkaana haluan ilmoittautua kurssille mihin aikaan tahansa minä päivänä tahansa." Hyvät käyttäjätarinat ovat neuvoteltavissa olevia, itsenäisiä ominaisuuksia, jotka ovat tilaajalle hyödyllisiä ja arvokkaita. Lisäksi niiden tulee olla riittävän pieniä ja niiden toteuttamisen vaatima työmäärä tulee olla arvioitavissa. Käyttäjätarinoissa esiteltyjen ominaisuuksien ei pidä olla liian subjektiivisia, vaan niiden tulee olla testattavissa. (Tieturi Oy 2010, 36, 38–39.) Käyttäjätarinat muodostavat tuotteen työlistaan kirjatut ominaisuudet. Työlista voidaan toteuttaa esimerkiksi soveltuvalla projektinhallintaohjelmistolla tai taulukkolaskentaohjelmalla. Esimerkki taulukkomuotoisesta tuotteen työlistasta löytyy liitteestä 1.

Tuotteen työlistaan kirjataan kunkin kohdan kuvaus, tärkeys ja aika-arvio. Tärkeyttä arvioitaessa otetaan huomioon kohtaan liittyvät riskit, sen tuottama arvo ja tarpeellisuus. Lisäksi jokaisesta ominaisuudesta voidaan kirjata hyväksymistestauksen läpäisykriteerit. (Schwaber & Sutherland 2010, 16–17.) Ominaisuus katsotaan valmiiksi vasta, kun se täyttää nämä valmiille ominaisuudelle asetetut kriteerit.

Tuotteen työlistaan kirjattavat ominaisuudet ja toiminnot eivät tule pelkästään asiakkaalta, vaan keneltä tahansa asianosaiselta: tuotteen käyttäjiltä, asiakaspalvelusta, markkinoinnin tai myynnin parissa työskenteleviltä sekä tuotteen toteuttamiseen osallistuvilta suunnittelijoilta ja kehittäjiltä. Tuoteomistajan tehtävänä on laittaa listan sisältämät asiat tärkeysjärjestykseen. (Schwaber & Beedle 2001, 7.) Lisäksi hän vastaa muutenkin tuotteen työlistan sisällöstä ja siitä, että lista on kaikkien asianosaisten saatavilla (Schwaber & Sutherland 2010, 16).

Mitä korkeammalle sijalle jokin tuotteen työlistan kohdista on määritelty, sitä tärkeämpi ja kiireellisempi se on. Kiireellisyydestä huolimatta korkean prioriteetin ominaisuuden toteutusta on syytä pohtia tarkasti. (Schwaber & Sutherland 2010, 16.) Tällaiset ominaisuudet vaikuttavat usein

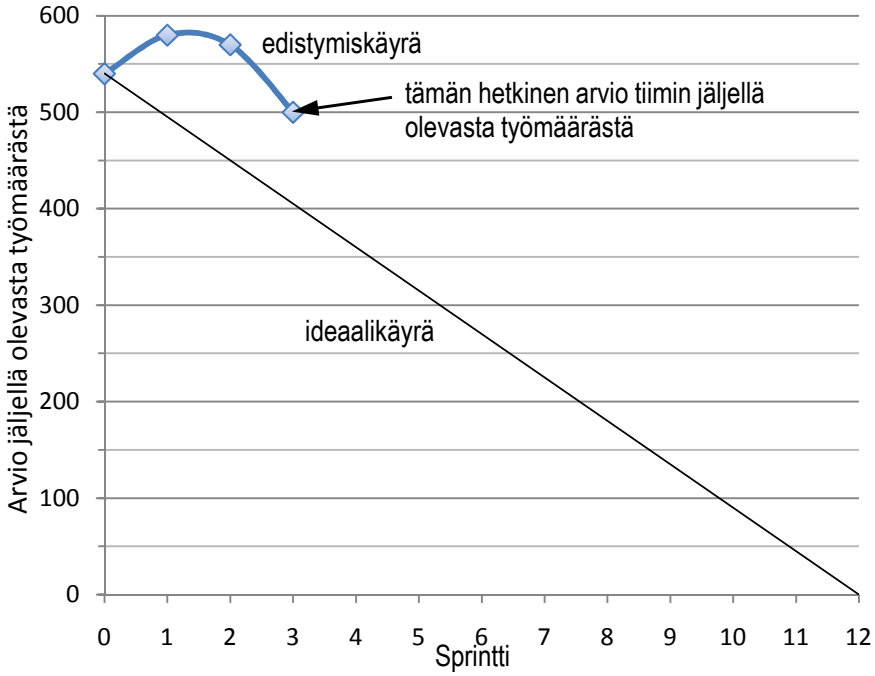
muihin tuotteen osiin, joten huolimaton suunnittelu voi johtaa ongelmiin matalamman prioriteetin ominaisuuksia toteutettaessa.

Tuotteen työlistan kohtia järjesteltäessä on teknisten ominaisuuksien lisäksi otettava huomioon taloudelliset seikat, erityisesti sijoitetun pääoman tuoton maksimointi (Deemer ym. 2010, 5). On tuoteomistajan tehtävä arvioida kohtien liiketaloudellista arvoa. Hänen tulee järjestää ominaisuuksia sen mukaan, miten nopeasti ne tuottavat rahaa tai auttavat tekemään säästöjä. Tällä parannetaan pääoman tuottoastetta ja siten liiketaloudellista kannattavuutta. (Virtanen 17.6.2010, luento.) Mikäli kyseessä on kaupallinen tuote, tuoteomistaja on vastuussa tuotteen avulla saatavasta tuotosta ja tappiosta. Jos kyseessä on kehittäjäyrityksen sisäinen tuote, tuoteomistajan tulee järjestää työlistaa korkeimman liiketoiminnallisen arvon ja matalimpien kustannusten perusteella. (Deemer ym. 2010, 5–6.)

Muutokset projektin toteutuksessa – esimerkiksi asiakkaan tilanteessa, teknologioissa ja kehitystiimin jäsenissä – aiheuttavat muutoksia myös tuotteen työlistalle. Tämän vuoksi matalan prioriteetin ominaisuuksia ei välttämättä kannata suunnitella yhtä yksityiskohtaisesti kuin korkean prioriteetin ominaisuuksia. Tällä vähennetään tarvetta tuotteen työlistan aikaavievälle uudelleentyöstämiselle. (Schwaber & Sutherland 2010, 17.)

3.5.2 Julkaisun edistymiskäyrä

Julkaisun edistymiskäyrän (Release Burndown Chart) avulla seurataan tuotteen työlistan jäljellä olevien ominaisuuksien työmäärää suhteessa käytettävissä olevaan aikaan. Työmäärää voidaan arvioida esimerkiksi tunteina tai päivinä, ja sitä verrataan käytettävissä olevien sprinttien määrään. Tuoteomistaja päivittää edistymiskäyrää jokaisen sprintin päätteeksi ja huolehtii siitä, että se on kaikkien tiimin jäsenten nähtävillä. (Schwaber & Sutherland 2010, 17–18.) Esimerkki julkaisun edistymiskäyrästä näkyy kuviossa 9.



KUVIO 9. Julkaisun edistymiskäyrä (Deemer, Benefield, Larman & Vodde 2010, 16)

Tuotteen työlistan kohtien aika-arviot määritellään samalla, kun suunnittelua tehdään. Aika-arvioita voidaan muuttaa tarvittaessa. Tuoteomistajan tehtävänä on selventää asioita, mutta aika-arvioiden määrittelystä vastaa julkaisun toteuttava kehitystiimi. (Schwaber & Sutherland 2010, 18.)

3.5.3 Sprintin tehtävälista

Kehitystiimin tuotteen työlistalta valitsemat, yhden sprintin aikana toteutettavat tehtävät muodostavat sprintin tehtävälistan (Sprint Backlog). Kunkin tehtävän tulee olla kestoltaan niin lyhyt, että se voidaan toteuttaa yhdessä päivässä tai jopa vähemmässä ajassa. Tällöin sprintin edistymistä voidaan seurata jokaisessa Päivän Scrum -palaverissa. (Schwaber & Sutherland 2010, 18.)

Kehitystiimi ylläpitää sprintin tehtävälistaa projektin edetessä. Kuten tuotteen työlistakin, myös sprintin tehtävälista voi muuttua projektin edetessä. Tiimi voi huomata, että sprintin lopputuloksen saavuttamiseksi tarvitaan enemmän tai vähemmän tehtäviä kuin alun perin suunniteltiin. Samaten voidaan havaita, että jonkin tehtävän toteuttamiseen kuluukin enemmän tai vähemmän aikaa kuin oletettiin. Kehitystiimillä on näiden havaintojen pohjalta oikeus lisätä uusia tehtäviä listalle tai poistaa tarpeettomiksi todettuja tehtäviä listalta. Myös aika-arvioita tulee päivittää jatkuvasti tehtävien edetessä ja valmistuessa. (Schwaber & Sutherland 2010, 18–19.)

Sprintin tehtävälistalle ei ole olemassa mitään tiettyä ulkoasua, vaan yritys voi valintansa mukaan hyödyntää esimerkiksi taulukkolaskentaohjelmaa, soveltuvaa projektinhallintaohjelmistoa tai vaikka liimattavia Post-it-paperilappuja. Tehtävälistaan päivitetään joka päivä jäljellä olevien työtuntien määrä. Nolla tarkoittaa, että kyseinen tehtävä on valmis. (Deemer ym. 2010, 10, 12.) Liitteestä 2 näkyy esimerkki taulukkomuotoisesta sprintin tehtävälistasta.

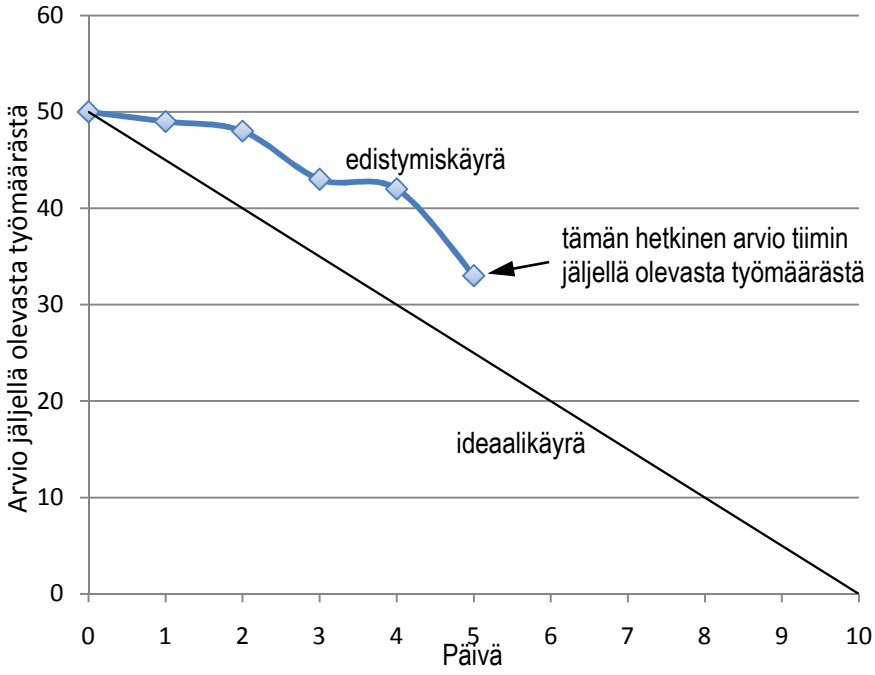
Ennen kuin sprintin tehtävälistalla oleva kohta voidaan luokitella valmiiksi, täytyy koko kehitystiimin tietää, mitä "valmis" tarkoittaa. Tiimin tulee sopia yhteinen käytäntö, jonka mukaisesti sprintin aikana tehty lopputulos voidaan määritellä valmiiksi. Tiimi voi esimerkiksi sopia, että tuotteen osa voidaan merkitä valmiiksi vasta, kun koodi on viimeistelty, yksikkötestattu, käännetty ja hyväksymistestattu. Tällöin vältytään väärinkäsityksiltä ja saadaan aikaiseksi julkaisukelpoinen tuotteen osa. Valmiin ominaisuuden kriteeriksi voidaan määritellä myös, että tarvittava dokumentaatio on kirjoitettu. (Schwaber & Sutherland 2010, 20.)

Tuoteomistajan tulee olla tietoinen tilanteesta, jossa kehitystiimi ei kykene tekemään kaikkea tarvittavaa. Tähän voi olla syynä esimerkiksi automaattisen testauksen puuttuminen. Tällöin sprintin aikana tehdyt osa-alueet luokitellaan joko valmiisiin tai keskeneräisiin. Valmiit ominaisuudet voidaan yhdistää aikaisemmin tehtyihin. Keskeneräinen työ puolestaan täytyy tehdä ennen kuin ominaisuus tai tuotteen osa voidaan ottaa käyttöön. Keskeneräiset työt lisätään tuotteen työlistalle ja niiden toteuttamiseen tarvittava aika otetaan huomioon julkaisun edistymiskäyrässä. Tämä menetelmä lisää avoimuutta, kun projektin todellinen tilanne tiedetään. (Schwaber & Sutherland 2010, 20–21.)

3.5.4 Sprintin edistymiskäyrä

Sprintin edistymiskäyrä (Sprint Burndown Chart) on julkaisun edistymiskäyrää vastaava työkalu, jolla ylemmän tason vaatimusten sijasta seurataan hienojakoisempia tehtäviä (Deemer ym. 2010, 15). Se havainnollistaa sprintin tehtävälistalla jäljellä olevien tehtävien työmäärää suhteessa käytettävissä olevaan aikaan. Edistymiskäyrä piirretään päivittäin laskemalla yhteen sprintin tehtävälistalla olevat aika-arviot. Tämä luku kuvaa jäljellä olevaa työmäärää. Näin kehitystiimi näkee konkreettisesti etenemisensä. Scrumin periaatteiden mukaisesti käytetyn ajan määrää ei seurata. Kiinnostuksen kohteena ovat vain jäljellä oleva työmäärä ja työpäivien määrä. (Schwaber & Sutherland 2010, 19.)

Sprintin edistymiskäyrän (kuvio 10) aikamääreenä käytetään yleensä päiviä. Edistymiskäyrän päätepiste kuvaa tiimin jäljellä olevaa tuntimääräistä työtä kyseisenä päivänä. Mikäli edistymiskäyrä kulkee jatkuvasti ideaalikäyrän yläpuolella, kehitystiimin kannattaa keskustella tuoteomistajan kanssa joidenkin työn alla olevien tuotteen työlistan ominaisuuksien siirtämisestä seuraavaan sprinttiin. Vastaavasti ideaalikäyrän alapuolella oleva edistymiskäyrä tarkoittaa sitä, että kehitystiimi ei ole valinnut riittävästi ominaisuuksia kuluvaan sprinttiin. (Tieturi Oy 2010, 74.)



KUVIO 10. Sprintin edistymiskäyrä (Deemer, Benefield, Larman & Vodde 2010, 13)

4 SCRUMIN KÄYTTÖÖNOTTO

Scrumin käyttöönotto vaatii aikaa ja vaivaa koko organisaatiolta. Se on empiirinen prosessi, jonka toteuttamiseen tarvitaan niin organisaation johdon kuin kaikkien muidenkin työntekijöiden panosta ja innostunutta osallistumista. Deemer ym. toteavat, että Scrumin käyttöönottoon voidaan päätyä yrityksen johdon aloitteesta. Koska Scrum pohjautuu itseohjautuviin tiimeihin, Scrumiin siirtymistä ei voida vain sanella yrityksen johdosta käsin. Tämän vuoksi parempi lähestymistapa onkin perehdyttää ohjelmistokehitystiimiä Scrumiin joko vertaistuen (muiden kehitystiimien) avulla tai johdon taholta, minkä jälkeen tiimi voi itsenäisesti päättää kokeilla käytäntöjä ennalta määritellyn jakson ajan. Saatuaan käytännön tuntumaa Scrumiin, tiimi voi arvioida kokemustaan ja päättää haluaako jatkaa Scrumin periaatteiden mukaisesti. (2010, 19.)

Leffingwell ja Smits kertovat julkaisussaan A CIO's Playbook for Adopting the Scrum Method of Achieving Software Agility, että kunhan yrityksessä on päätetty ottaa Scrum käyttöön, voidaan prosessin lopputuloksena odottaa entistä tehokkaampaa ohjelmistokehitystä ja nopeammin reagoivaa, kilpailukykyisempää yritystä. Lisäksi Scrumin onnistuneesta käyttöönotosta voidaan saada seuraavia hyötyjä: tuotteiden kehitystahti nopeutuu, loppukäyttäjät saavat enemmän arvoa, tuotteet ovat parempilaatuisia, kehitykseen liittyvät riskit ovat pienempiä, käyttäjätyytyväisyys kasvaa ja yrityksen työntekijöiden moraali paranee. (2005, 10, 22.)

Schwaberin mukaan Scrumin käyttöönotto vaatii suurta vaivaa kaikilta siihen osallistuvilta. Yrityksen tulisi aluksi vakuuttua siitä, että vaivannäkö todella kannattaa. Johdon tulee nähdä konkreettisia ongelmia, jotka heidän mielestään voivat olla ratkaistavissa Scrumin avulla. Vasta tämän jälkeen resursseja ja aikaa vaativaan käyttöönottoprosessiin kannattaa ryhtyä. (2007, 5.)

Ohjelmiston vaatimukset voivat muuttua nopeasti projektin edetessä, jolloin tarvitaan mukautumiskykyistä ohjelmistokehitysprojektimenetelmää. Leffingwellin ja Smitsin mukaan Scrumin käyttöönotosta vastaavan johtajan tuleekin ymmärtää, että empiirinen, kokemusperäinen, prosessi sopii paremmin tehokkaaseen ohjelmistokehitysprojektiin kuin suunniteltu prosessi. Scrumiin liittyy käsitys siitä, että itseorganisoituva ja itseohjautuva tiimi tuottaa parempia ohjelmistoja kuin muulla tavalla toimiva tiimi. Johtajan tulee ymmärtää, että hän voi odottaa laadultaan ja arvoltaan hyvää ohjelmistoa suunnitellun aikataulun ja budjetin rajoissa, mutta hän ei kuitenkaan voi tietää, millaisia toiminnallisuuksia tiimi loppujen lopuksi toimittaa. (2005, 5.)

Edellä mainitut perusperiaatteet voivat vaatia merkittäviäkin muutoksia Scrumin käyttöönottavassa organisaatiossa. Kuitenkin vain näiden perusperiaatteiden ymmärtämisen ja hyväksymisen kautta voidaan varmistaa, että Scrumilla toteutetun ohjelmistokehityksen tiellä ei ole esteitä. (Leffingwell & Smits 2005, 5.)

Kuten luvussa 3.3.3 on kerrottu, Scrumiin kuuluu oleellisesti kehitystiimi, joka toimii sekä itseohjautuvasti että vastuunalaisesti. Leffingwellin ja Smithsin mukaan tämä toimintatapa tuo muutoksia useisiin organisaatioihin, joissa ei ole käytetty Scrumia, sillä Scrumissa ei käytetä hierarkkista, teknistä, johtajavetoista, ohjeisiin nojaavaa lähestymistapaa. Scrumia käyttöönotettaessa on hyväksyttävä, että vain tuoteomistaja on vastuussa päämäärien ja niiden tärkeysjärjestyksen määrittelyssä ja että kehitystiimin on saatava työskennellä itsenäisesti asetettujen tavoitteiden saavuttamiseksi. (2005, 6.)

Ensimmäisten viidentoista vuoden aikana Scrumin käyttöönotto tapahtui tyypillisesti alhaalta ylöspäin eli projektitiimi halusi toimia Scrumin periaatteiden mukaisesti. Viime aikoina organisaatioissa on alettu toteuttaa muutosta ylhäältä alaspäin, osana johtamisen toimintatapoja. Tällä on pyritty nopeuttamaan yrityksen reagointikykyä ja tehostamaan tuottavuutta. Tämän käyttöönottotavan onnistuminen edellyttää kuitenkin huolellista harkitsemista ja valmistelua. (Leffingwell & Smits 2005, 8.)

Leffingwell ja Smits kertovat artikkelissaan sekä ohjelmistokehitysprosessin että koko organisaation muuttamisesta Scrumin periaatteiden mukaiseksi. Heidän mukaansa organisaation täytyy ottaa vastuu sekä kehitystyön tekemisestä Scrumin periaatteiden mukaisesti että eteen tulevien kehitystyötä estävien tekijöiden poistamisesta. Näistä ensimmäinen parantaa ohjelmistojen valmistumista ja jälkimmäinen sijoitetun pääoman tuottoa. Molemmat asiat ovat haastavia ja vaativat kovaa työtä varsinaisen ohjelmiston kehitystyön lisäksi. (2005, 8.)

Kuten aikaisemmin on kuvattu, Scrum-mestarin tehtävänä on huolehtia, että kehitystiimillä on kaikki edellytykset ja mahdollisuudet hyvään työskentelyyn. Organisaatiotasolla vastaavanlaista tehtävää hoitaa yrityksen johtaja, yrittäjä tai esimerkiksi tietohallintojohtaja, jota Leffingwell ja Smits kutsuvat organisaatiotason Scrum-mestariksi. Hänen tehtävänään on työskennellä varsinaisen Scrum-tiimin ulkopuolella, ja huomioida, tunnistaa sekä poistaa kaikki organisaatiotason esteet, jotka voivat estää ketterän kehitystavan onnistumisen. Juuri organisaatiotason Scrum-

mestari on avainasemassa siinä, onnistuuko vai epäonnistuuko käyttöönottoprosessi. (Leffingwell & Smits 2005, 8, 22.)

Scrumin täydellinen omaksuminen voi viedä organisaatiolta jopa kaksi vuotta. Scrumin käyttöönottoon kuluvaa aikaa ei voida kiirehtiä edes johdon innokkuudella tai sitoutumisella, sillä prosessin ytimenä on organisaation muuttuminen omalla tahdillaan. Scrumin käyttöönoton nopeus on suoraan kytköksissä kolmeen asiaan: organisaatiossa tarvittavan muutoksen laajuuteen, organisaatiossa tarvittavan ohjelmistokehityksen ja toimitusvarmuuden parantamisen kiireellisyyteen ja organisaation johdon tehokkuuteen. (Leffingwell & Smits 2005, 8.)

Leffingwellin ja Smitsin mukaan Scrumin käyttöönottoa ei voida suunnitella ja panna toimeen tehtävälistoilla, proseduureilla ja lomakkeilla. Scrumin yhtenä päätarkoituksena on tunnistaa organisaatiossa sellaiset tekijät, jotka estävät ohjelmistojen parhaimman mahdollisen luomisen. Koska jokainen organisaatio on erilainen, Scrumin käyttöönoton toteuttaminenkin on eri organisaatioissa erilaista. Monet käyttöönottoprosessin aikana havaittavista, muutosta vaativista asioista voivat olla organisaatiolle luontaisia, joten niiden muuttaminen voi olla vaikeaa. Huolellisella etukäteissuunnittelullakaan ei voida ohittaa kaikkia väistämättä kohdattavia vaikeuksia. Käyttöönoton onnistumisessa ehdottoman tärkeää on, että johto on olennaisesti mukana esteiden havainnoimisessa ja poistamisessa. Scrumin käyttöönotosta huolehtivalta organisaatiotason Scrum-mestarilta vaaditaan sitä, että hän omaksuu Scrumin periaatteet ja ryhtyy ohjaamaan muutosta määrätietoisesti. (2005, 9.)

Joka tapauksessa ennen kuin yrityksen laajuiseen käyttöönottoon ryhdytään, kannattaa Scrumia kokeilla pilottiprojektissa tai useammassakin. Vaikka Scrumin käytännöt ovat periaatteessa lukemalla omaksuttavia, Scrum-mestarin ja pilottiprojektin kehitystiimin varhaisessa vaiheessa tapahtuva kouluttaminen voi olla suureksi avuksi, jotta kokeilun onnistumisella on yleensä mahdollisuuksia. (Schwaber 2007, 5.)

Pilottiprojekti käynnistetään muodostamalla Scrum-tiimi, jonka jäseniä koulutetaan riittävässä määrin. Sitten järjestetään Scrumin prosessikuvauksesta tuttu sprintin suunnittelukokous. Tuotteen työlistalle valitaan joitakin ohjelmiston tärkeitä ominaisuuksia. Tiimin muodostuminen ja heidän ymmärryksensä kehitettävästä järjestelmästä tapahtuu vähitellen usean sprintin aikana, joten pilottiprojektissa on toteutettava vähintään kolme sprinttiä. Kolmannen sprintin jälkeen voidaan jo arvioida koko projektin kustannuksia saatujen kokemusten perusteella. Tällöin

Schwaberin mukaan on mahdollista havaita Scrumin tuoma tehokkuus ja arvo, ja saada käsitys Scrum-projektien kulusta ja Scrumin käyttöön siirtymiseen liittyvistä muutostarpeista. Ensimmäisen pilottiprojektin jälkeen Scrumia voidaan testata toisessa projektissa, sellaisessa, joka on astetta vaativampi tai jonka toteuttamisessa yrityksellä on ollut ongelmia. (Schwaber 2007, 5.)

Mikäli yrityksessä päätetään pilottiprojektin perusteella ryhtyä viemään Scrumin käyttöönottoa eteenpäin, on hyvä tietää muutoksista, joita muut yritykset ovat käyneet läpi käyttöönoton yhteydessä (Schwaber 2007, 5). Scrumin käyttöönottoon liittyviä haasteita käsitellään tarkemmin luvussa 4.2.

4.1 Käyttöönoton vaiheet

Leffingwell ja Smits käsittelevät artikkelissaan Scrumin käyttöönottoprosessia pilottiprojektista aina organisaatiotason omaksumiseen asti. He kertovat tyypillisiä tapoja, joiden avulla organisaatiotason Scrum-mestari voi lähteä viemään Scrumin käyttöönottoa eteenpäin. He ovat muodostaneet kuusivaiheisen "valmentajan oppaan" (Playbook), joka antaa näkemyksiä tavoista, joilla tarvittava muutos voidaan saada aikaan. (2005, 10.)

4.1.1 Yleiskatsaus, arviointi ja pilottiprojektin valmisteleminen

Valmentajan oppaan ensimmäinen vaihe sisältää yleiskuvan ja tilannearvion muodostamisen sekä pilottivaiheen valmistelun. Ensiksi arvioidaan organisaation valmiusaste ketteryyteen ja järjestetään koulutus ensivaiheen osallistujille. Lisäksi luodaan tuotteen työlistat pilottiprojektille. Organisaation johdon tulee organisaatiotason Scrum-mestarin opastuksella käyttää aluksi aikaa sen pohtimiseen, minkälaisia muutoksia Scrumin myötä tulee tapahtumaan ja halutaanko käyttöönottoprosessiin todella ryhtyä. Mikäli käyttöönottoon päätetään ryhtyä ja yleisistä suuntaviivoista on päästy selvyyteen, järjestetään koko organisaatiolle Scrumin esittelytilaisuus, jossa kasvatetaan yleistä tietoutta Scrumista ja sen käsitteistä. (Leffingwell & Smits 2005, 10.)

Tämän jälkeen organisaatiotason Scrum-mestarin tulee arvioida organisaation valmiutta muutokseen ja toteuttaa seuraavat toimet: yhden tai useamman pilottiprojektin valitseminen, koulutuksen aikatauluttaminen ja pilottiprojektin resursoiminen. Tähän ensimmäiseen vaiheeseen käytetään Leffingwellin ja Smitsin ohjeistuksen mukaisesti kaksi päivää. Tarvittaessa tukena voidaan käyttää ulkopuolista asiantuntijaa. Pilottiprojektin Scrum-mestarille tulee järjestää parin päivän

mittainen koulutus ja tuoteomistajalle oma päivänmittainen koulutuksensa, jotta ensiksi mainittu saa riittävästi tietoutta pilottiprojektin eteenpäin viemisestä ja jälkimmäinen sijoitetun pääoman tuoton maksimoimisesta Scrumin avulla. (Leffingwell & Smits 2005, 10.)

Tässä vaiheessa johdon tulee arvioida ja muodostaa mittarit, joilla Scrumin käytön onnistumista organisaatiossa voidaan valvoa. Lisäksi johdon tulee määritellä se arvo, jolla pilottiprojekti voidaan katsoa onnistuneeksi. Tähän tulisi käyttää aikaa noin viikko. Tässäkin vaiheessa voi olla hyväksi hyödyntää ulkopuolisia Scrum-asiantuntijoita määrittelyissä. Ensimmäisen vaiheen lopuksi käytetään päivä sellaisten työlistojen laatimiseen, joilla voidaan havainnoida organisaatiossa tapahtuvia muutoksia. Niiden avulla voidaan jäljittää ja arvioida esteet, joita pilottiprojektien aikana nousee esille. (Leffingwell & Smits 2005, 11.)

4.1.2 Pilottiprojekti

"Valmentajan oppaan" toinen vaihe liittyy pilottiprojektiin. Pilottiprojektin tavoitteena on käydä läpi Scrumin eri vaiheet siten, että organisaatiolle saadaan esiteltyä parantuneen ketteryyden tuomat hyödyt. Tarvittaessa toteutetaan useampia pilottiprojekteja, joiden avulla Scrum-osaamista ja -tietoisuutta saadaan syvennettyä. Sekä pilottiprojektiin valitun Scrum-mestarin että organisaatiotason Scrum-mestarin tulee seurata tiiviisti pilottiprojektin edistymistä, jotta sekä Scrumin käyttöön liittyvät että organisatoriset esteet saadaan tunnistettua saman tien niiden ilmaannuttua. Näin esteisiin voidaan joko puuttua heti, mikäli mahdollista, tai ne voidaan merkitä ylös organisaation muutoksia vaativalle työlistalle myöhempää tarkastelua varten. (Leffingwell & Smits 2005, 11.)

Hyvä kestoaika pilottiprojektille on 3–6 iteraatiota eli 3–6 kuukautta. Projektin jälkeen järjestetään kahden päivän mittainen katselmus, jossa arvioidaan pilottiprojektin saavutuksia ja esteitä: mikä onnistui hyvin ja mikä vaatii vielä kehittämistä. Myös sijoitetun pääoman tuottoa tulee arvioida, samoin projektin vaikutusta liiketoimintaan sekä sisäisiin suhteisiin ja asiakassuhteisiin. Lopuksi käytetään yksi päivä Scrumin käyttöönottoprosessiin liittyvien jatkosuunnitelmien tekemiseen. Huolimatta siitä, että Scrum-mestarilla on suurin vastuu pilotin toteuttamisesta, voidaan tässäkin vaiheessa käyttää apuna ulkopuolista asiantuntijaa. (Leffingwell & Smits 2005, 11.)

4.1.3 Organisaatiotason laajentuminen

Kun pilottiprojekti ja sitä mahdollisesti seuraavat syventävät pilottiprojektit on saatu onnistuneesti valmiiksi, voidaan Scrumin ja sen tuomien hyötyjen käyttöä laajentaa muuhun organisaatioon. Tähän mennessä pitäisi olla jo muodostunut kuva hyödyllisistä käytännöistä, laajemman omaksumisen tiellä olevista esteistä ja siitä, millä alueilla tarvitaan vielä lisäkoulutusta. Tässä vaiheessa voidaan miettiä, saataisiinko jonkin muun ketterän menetelmän käytännön soveltamisesta lisähyötyä. Voidaan pohtia, olisiko esimerkiksi Extreme Programming -menetelmään kuuluvista ohjelmointitavoista etua. (Leffingwell & Smits 2005, 11–12.)

Pilottiprojektissa mukana ollutta Scrum-mestaria voidaan jatkokouluttaa ja tarvittaessa voidaan aloittaa yhden tai useamman uuden Scrum-mestarin kouluttaminen. Myös kehitystiimin jäseniä on syytä harjaannuttaa edelleen, jotta he voivat toimia entistä paremmin ketterien ohjelmistokehitysprojektien edellyttämillä tavoilla. Jotta organisaation jäsenillä on yhteinen kieli ja toimintamalli, Scrum-tiimin ulkopuolisiakin työntekijöitä varten on syytä järjestää 2–4 tunnin koulutustilaisuuksia. Scrumin käyttöönoton hyväksymisen ja avoimuuden parantamiseksi henkilöstölle voidaan antaa Scrumista kertovia artikkeleita tai kirjoja luettavaksi. On myös tärkeää, että pilottiprojektien tulokset ovat kaikkien nähtävillä. Lisäksi voidaan järjestää esimerkiksi johdon ja työntekijöiden yhteisiä lounaita tai tunnin–kahden mittaisia seminaareja, joissa käsitellään avoimesti niitä muutoksia, joita organisaatiossa on tapahtumassa. (Leffingwell & Smits 2005, 11–12.)

4.1.4 Vaikutuksen saavuttaminen

Kun organisaatiolla on riittävästi täsmällistä ja hiljaista tietoa Scrumista, ollaan valmiita siirtymään neljänteen vaiheeseen. Kehitystiimi jatkaa edelleen työskentelynsä kehittämistä ja organisaatiotason Scrum-mestari vastaa esteiden eliminoimisesta yhdessä asianosaisten työntekijöiden kanssa. Jokaisen sprintin jälkeen arvioidaan tuloksia laadullisilla ja määrällisillä mittareilla. (Leffingwell & Smits 2005, 12–13.)

Scrumia käyttävät projektit seuraavat toisiaan, ja organisaation koon salliessa useita Scrum-projekteja toteutetaan rinnakkain. Vaikka tiimin jäsenten vaihtuminen vaikuttaa aina hetkellisesti tehokkuuden vähenemiseen, on tärkeää aika-ajoin kierrättää eri projektien tiimiläisiä projektista toiseen. Aluksi tiimin jäseniä vaihdellaan esimerkiksi yhden–kahden vuoden välein, myöhemmin

tarvittaessa. Näin saadaan esille uusia näkökulmia ja samalla havaittuja esteitä voidaan purkaa. (Leffingwell & Smits 2005, 12–13.)

4.1.5 Mittaus, arviointi ja mukauttaminen

Viidenteen vaiheeseen mentäessä suurin osa organisaatiosta toimii ketterien periaatteiden mukaisesti. Pääasiallisena tehokkuuden mittarina toimii kunkin sprintin lopputuloksen onnistuneisuus. Näin voidaan arvioida miten Scrum-tiimi on onnistunut uudessa toimintatavassa. Toimivien ohjelmiston osien riittävän usein tapahtuvat toimittaminen on tärkeintä, mutta organisaatiotasolla tarvitaan myös muita mittareita. (Leffingwell & Smits 2005, 13.)

Leffingwell ja Smits jakavat organisaatiotason mittarit kahteen luokkaan: prosessimittareihin ja projektimittareihin. Prosessitasolla Scrum-tiimien ja koko organisaation edistymistä Scrumin periaatteiden omaksumisessa voidaan arvioida laadullisin keinoin, kuten miten määrätietoisesti Scrum-tiimi on ylläpitänyt tuotteen työlistaa, ja miten tehokkaasti Scrum-prosessille tärkeitä vaiheita (esimerkiksi Päivän Scrumia tai sprintin suunnittelukokouksia) on hyödynnetty. Jonkin tietyn Scrum-tiimin ja sen vastuulla olevan tuotteen tai palvelun projektitasoista edistymistä voidaan seurata niin perinteisillä mittareilla kuin erityisillä Scrumiin liittyvillä mittareilla. Perinteisiä mittareita ovat esimerkiksi löytyneiden virheiden määrä sekä sovitulla tavalla testatun koodin määrä. Scrumin omia mittareita ovat esimerkiksi sen arvioiminen, onko tuotteen työlista riittävän kuvaava ja onko se järjestelty hyvin sekä järjestetäänkö Päivän Scrumit ajallaan ja osallistuvatko kaikki asianosaiset niihin. (Leffingwell & Smits 2005, 13.) Leffingwellin ja Smitsin artikkeliin liitetyssä, Hartmannin ja Stallingsin kehittämässä pohjassa on esitelty Scrum-prosessiin liittyviä mittareita, joita voidaan hyödyntää Scrumin käyttöönoton aikana. Tämä pohja löytyy liitteestä 3.

4.1.6 Lopullinen laajentaminen

Mikäli Scrum halutaan ottaa käyttöön koko organisaatiossa, niin tämä laajentuminen toteutetaan "Valmentajan oppaan" esittelemässä viimeisessä vaiheessa. Laajentaminen tehdään sopivaksi katsotuissa erissä työntekijöiden määrästä riippuen, ja viimeistä vaihetta jatketaan niin kauan, kunnes kaikki organisaation jäsenet työskentelevät ketterästi. Tässä vaiheessa olemassa olevia käytäntöjä hiotaan entisestään ja opittuja asioita jaetaan tiimien välillä. Saadun kokemuksen avulla Scrumin periaatteita voidaan tarvittaessa muokata paremmin organisaation tarpeita vastaaviksi. (Leffingwell & Smits 2005, 14.)

Laajentumisen yhteydessä asiakkaiden roolia Scrum-projekteissa vahvistetaan. Järjestämällä tuoteomistajien tehtäviin liittyviä koulutuksia, heidän edustajiaan voidaan kutsua osallistumaan projekteihin tiiviimmin. Scrumin tutki ja mukauta -periaatetta hyödynnetään prosessien ja käytäntöjen jatkokehityksessä. Tähän vaiheeseen päästessään organisaatio saa täyden hyödyn Scrumin tuottavuutta, liiketoimintaa ja organisaatiokulttuuria parantavista eduista. Näin ollen Scrumin käytäntöjen laajentuminen koko organisaation tasolle on ideaalitilanne. (Leffingwell & Smits 2005, 14.)

4.2 Scrumin käyttöönottoon liittyvät haasteet

Scrumin käyttöönoton yhteydessä yritys ja sen työntekijät kohtaavat monenlaisia haasteita ja ongelmia. Nämä haasteet pitää kohdata avoimin mielin, ja ne pitää ratkaista rakentavasti, jotta käyttöönotto voi ylipäätään onnistua. Jokainen käyttöönottoprosessi on erilainen, sillä yritykset ovat erilaisia. Tämän vuoksi on mahdotonta käsitellä aihetta kattavasti, mutta tässä luvussa nostetaan esiin lähdeaineistossa esiteltyjä tyypillisiä haaste- ja ongelmatilanteita.

Leffingwellin ja Smitsin esille tuomat haasteet liittyvät Scrum-prosessiin, ihmistenvälisiin käytäntöihin, tuotteen kehittämiskäytäntöihin tai organisaatioon itseensä. Erilaiset ongelmat Scrum-prosessin toteuttamisessa voivat heikentää Scrumin tehokkuutta. Ihmistenvälisiin käytäntöihin liittyvät epäkohdat voivat estää kehitettävän tuotteen kehitystä, jakelua, tukitoimintoja ja käyttöä. Näihin ongelmiin puuttumalla voidaan huolehtia siitä, että jokainen kehitystyöhön osallistuva on parhaalla mahdollisella tavalla mukana projektissa. Ongelmat tuotteen kehittämiskäytännöissä puolestaan heikentävät esimerkiksi tuotteeseen investoidun pääoman tuottoa tai tuotetta itseään. Kehitystiimin ulkopuoliseen organisaatioon liittyvät ongelmat taas saattavat estää tiimiä toimittamasta kehitettävää tuotetta tai sen osaa riittävän nopeassa aikataulussa. (2005, 15.)

Näihin eri tavoilla ja tasoilla ilmeneviin ongelmiin tulee puuttua, mutta aina ei ole itsestään selvää, kenen tai keiden tulisi vastata asiasta. Tämän vuoksi onkin hyvä jo etukäteen miettiä, mitä ongelmia Scrumin käyttöönotossa ja käytössä voi mahdollisesti ilmetä, ja keiden pitäisi puuttua ongelmiin ja auttaa niiden ratkaisemisessa. Myös kustannuskysymykset tulee ottaa huomioon. (Leffingwell & Smits 2005, 15.)

Deemer, Benefield, Larman ja Vodde kertovat julkaisussaan, että Scrumia hyödynnettäessä on hyväksyttävä se, ettei menetelmä ratkaise kehitysongelmia, vaan ainoastaan paljastaa ne.

Tämän etuna on mahdollisuus ongelmiin puuttumiseen jo varhaisessa vaiheessa. Ensimmäinen sprintti on yleensä erittäin haastava Scrum-tiimille, mutta jo sen lopussa voidaan usein nähdä Scrumin tuomia etuja verrattuna aikaisempiin käytäntöihin. Kehitystiimi voi kokea epäonnistuneensa, mikäli se ensimmäisen sprintin aikana epäonnistuu asettamiensa tavoitteiden saavuttamisessa. Tämä voi johtua esimerkiksi puutteellisesta tehtävien analysoinnista ja niihin tarvittavan ajan arviointivaikeuksista. Tätä ei tulisi kuitenkaan pitää epäonnistumisena, koska juuri tällaiset kokemukset ovat välttämättömiä askeleita kohti realistisempaa ja harkitsevampaa vaatimusten asettamiskykyä. (2010, 19.)

Scrumin perusperiaatteiden muuttaminen ja niistä joustaminen törmättäessä hankaluuksiin ovat yleisiä virheitä Scrumin käyttöönotossa. Kokematon tiimi voi esimerkiksi haluta pidentää sprintin kestoa voidakseen saavuttaa sprintille asettamansa tavoitteet. Tällöin tiimi ei kuitenkaan tule oppimaan miten arvioida ja hallita ajankäyttöä. Mikäli Scrum-mestari tällöin ei valmenna ja tue kehitystiimiä, voidaan päätyä tilanteeseen, jossa Scrum vain heijastaa organisaation heikkouksia ja toimintahäiriöitä. Tämä heikentää niitä todellisia etuja, joita Scrum tarjoaa: hyvien ja huonojen käytäntöjen näkyville nostamista ja organisaatiolle tarjottua mahdollisuutta kohottaa toimintansa uudelle, tehokkaammalle tasolle. (Deemer ym. 2010, 19.) Kehitystiimi siis vastaa kehitettävistä toiminnallisuuksista asetettujen käytäntöjen ja organisaation standardien puitteissa. Scrum-mestarin tehtäviin taas kuuluu vastata prosessista ja kehitystoimintaa haittaavien esteiden poistamisesta. Hän voi auttaa tiimiä pysymään oikeissa toimintarajoissa esittämällä kysymyksiä ja tarjoamalla neuvoja. (Schwaber 2004, 110.)

Scrumin käyttöönotossa tulee olla kärsivällinen ja varata sille riittävästi aikaa. Schwaberin mukaan ajanjakso kolmannesta yhdeksänteen kuukauteen on usein erityisen haasteellinen. Tässä vaiheessa yrityksessä esiintyneet ongelmat ja toimintahäiriöt tulevat päivänvaloon. Niihin ei ole puututtu aikaisemmin, koska se on koettu vaikeaksi tai niiden olemassaolo on vain hyväksytty. Nyt niihin kuitenkin joudutaan kiinnittämään huomiota, mikä voi tuntua raskaalta. Tällöin täytyy muistaa miettiä Scrumin käyttöönoton myötä jo aikaansaatuja parannuksia ja jatkaa muutosprosessin eteenpäin viemistä. (2007, 6.)

Liitteen 4 taulukossa on listattu joitakin esimerkkejä haittatekijöistä ja ongelmista, joita Scrumia käyttöönottavissa organisaatioissa on havaittu. Tämä lista voi auttaa alkuun mahdollisesti esiin nousevien haittojen havaitsemisessa ja toimii myös ennakoivana varoituksena siitä, että näitä ongelmia luultavasti ilmenee jossain vaiheessa käyttöönottoprosessia. On kuitenkin muistettava,

että sekä jokainen yritys ja organisaatio että käyttöönottoprosessi on erilainen, joten eteen tulee varmasti myös muita, listan ulkopuolisia ongelmatilanteita. (Leffingwell & Smits 2005, 15.)

4.2.1 Käyttöönotosta toteutetun tutkimuksen esittely

Moe, Dingsøyr ja Dybå tutkivat yhdeksän kuukauden ajan ohjelmistokehitysyritystä, jossa Scrumin hyödyntäminen laitettiin alulle. Suurin osa prosessiin osallistuvista ohjelmistokehittäjistä oli aikaisemmin tottunut työskentelemään itsenäisesti moduuleihin jaettujen projektien parissa, joten heille tämä oli ensimmäinen kokemus suuremmassa projektissa työskentelemisestä. Tutkimus toteutettiin kenttätutkimuksen ja haastatteluiden avulla käyttöönoton suunnitteluvaiheen ja kuuden sprintin aikana. Tutkijat havaitsivat, että siirtyminen yksilöitä korostavasta työtavasta itseohjautuviin tiimeihin vaati uudelleenorientoitumista, ei ainoastaan ohjelmistokehittäjiltä vaan myös johdolta. (2009, 1, 11.)

Kahden ensimmäisen sprintin aikana Scrum-tiimi koki vanhoista työtavoista luopumisen haasteellisena. Kukin tiimin jäsenistä työskenteli tietyn ohjelmistomoduulin parissa alusta loppuun asti, ja kehittäjät loivat tyypillisesti oman suunnitelmansa vastuullaan oleville moduuleille. Usein tämä tapahtui keskustelematta asiasta muiden tiimin jäsenten kanssa. Scrum-mestari ei puuttunut tähän toimintatapaan vaan antoi sen ohjelmistokehittäjän, joka tiesi tietystä aiheesta eniten, työskennellä kyseisen ominaisuuden parissa. Hänen mielestään tuottavuuden kannalta ei ollut realistista olettaa, että useat ihmiset voisivat työskennellä saman asian parissa. (Moe ym. 2009, 6–7.)

Tutkimuksessa kävi myös ilmi, että ohjelmistokehittäjät priorisoivat yksilöllisiä tavoitteitaan tiimin tavoitteiden edelle. Osa tiimin jäsenistä oli motivoitunut täydellisen teknisen ratkaisun löytämiseen sen sijaan, että ajattelisi osa-alueen toteuttamiseen varattua aikaa. Yksi kehittäjistä toteutti muille kertomatta ominaisuuksia, joita tarvittaisiin mahdollisesti vasta tulevaisuudessa. Hän käytti tähän runsaasti aikaa eikä suostunut hyväksymään aikaisemmin tehtyä päätöstä siitä, että kyseinen ominaisuus otetaan toteutettavaksi vasta myöhemmin. Tästä johtuen Scrum-mestari menetti luottamuksensa kyseiseen tiimin jäseneen ja alkoi valvoa hänen toimintaansa. Tämän seurauksena häntä ei enää huomioitu tasavertaisena tiimin jäsenenä. (Moe ym. 2009, 7.)

Scrum-mestari huomasi, että tiimi ei raportoinut hänelle kohtaamiaan ongelmia. Tiimin jäsenet kokivat, että Scrum-mestari ylireagoi Päivän Scrumeissa kerrottuihin ongelmiin, minkä vuoksi tiimi ei enää halunnut käsitellä ongelmia hänen läsnä ollessaan. Tilanne parani, kun Scrum-mestari

keskusteli asiasta rakentavasti tiimin kanssa. Hankaluudeksi muodostui kuitenkin se, että ongelmat havaittiin myöhäisessä vaiheessa ja niiden nähtiin olevan henkilökohtaisia. Tähän oli syynä se, että tiimin jäsenet työskentelivät omien vastuualueidensa parissa yksin ja itsenäisesti, jolloin he eivät saaneet palautetta työstään. Tiimiläiset keskittyivät työskentelemään omien moduuliensa parissa eivätkä pyytäneet tukea toisilta, vaikka olisivat sitä tarvinneet. He kuvittelivat voivansa ratkaista ongelmat itsenäisesti ennen seuraavaa Päivän Scrumia, vaikka usein näin ei kuitenkaan käynyt. (Moe ym. 2009, 7.)

Päivän Scrumissa ei aina kiinnitetty huomiota toisten tiimiläisten sanomisiin, koska esiin nostetut asiat eivät koskettaneet kuin kulloinkin äänessä olevaa tiimin jäsentä. Tämä koettiin ongelmallisena projektin kannalta: yksilöllistä ja itsenäistä työtä korostava tapa heikensi tiimin jäsenten välistä vuorovaikutusta, palautteen antamista ja saamista sekä mahdollisuutta seurata toisten aikaansaannoksia. (Moe ym. 2009, 7.)

Kommunikaatio-ongelmia ilmeni myös tilanteessa, jossa yksi tiimin jäsenistä kehitti itsenäisesti tietokannan, jota käytettäisiin koko projektissa. Hän selitti tietokantamoduulin rakenteen muille tiimin jäsenille Päivän Scrumissa ennen lomalle jäämistään mutta ei varmistanut sitä, olivatko tiimikaverit todella ymmärtäneet miten uusi ominaisuus toimii. Muilla tiimiläisillä oli moduuliin liittyviä ymmärtämisvaikeuksia, ja he päättivät kirjoittaa uudelleen kyseiseen moduuliin liittyvän koodin käyttäen siihen yli sata työtuntia. Tästä ja muista kahden ensimmäisen sprintin aikana havaituista haasteellisista tilanteista viisastuneina tiimi päätti toisen sprintin jälkitarkastelussa kiinnittää jatkossa huomiota siihen, että kukaan ei yksin vastaa jostain osa-alueesta ja kommunikaatioon kiinnitetään huomiota. (Moe ym. 2009, 7.)

Kolmanteen sprinttiin siirryttäessä kehitystiimi oli alkanut tottua sprinttien suunnitteluun ja ohjaamiseen, mikä lisäsi kehittäjien motivaatiota ja tyytyväisyyttä. Monilla tiimin jäsenillä oli kuitenkin vaikeuksia sprintin työlistalla olevien ominaisuuksien muuttamisessa varsinaisiksi työtehtäviksi. He olivat tottuneet vapaasti muotoiltujen ominaisuuksien sijasta tarkemmin spesifioituihin teknisiin vaatimuksiin. Lisäksi tiimiläiset kokivat, että kukaan ei oikeasti tiedä mitä he ovat tekemässä. Tämä ilmaisee sitä, että tiimille ei ollut muodostunut selkeää näkemystä siitä, miten lopputuloksiin voidaan päästä. Tiimi ei aidosti johtanut itseään vaan pyrki tukeutumaan liiaksi vanhoihin käytäntöihin. (Moe ym. 2009, 7–8.)

Kolmannen ja neljännen sprintin aikana tiimin sisäinen kommunikaatio ja palautteen antaminen parani. Scrumin noudattaminen pakottaa tiimin jäsenet työskentelemään tiiviisti yhdessä ja muistuttaa keskustelun tärkeydestä. Tiimi ei kuitenkaan aina päässyt yhteisymmärrykseen esimerkiksi teknisistä asioista. Tällöin saatettiin päätyä tilanteeseen, jossa toiset ryhmän jäsenet jyräsivät mielipiteillään muiden ideat, ja ihmiset jatkoivat itsenäistä työskentelyä. (Moe ym. 2009, 8.) Aidon vuorovaikutuksen toteuttaminen ja muiden kuunteleminen onkin erittäin tärkeää projektin onnistumisen kannalta.

Päivän Scrumien toteutuksessa esiintyi edelleen vaikeuksia. Tiimin jäsenet kokivat nämä palaverit lähinnä tilanteina, joissa Scrum-mestarille raportoitiin mitä on saatu aikaan ja mitä tehdään seuraavaksi. Tiimiläiset olivat huomanneet, että kun Scrum-mestari ei syystä tai toisesta osallistunut Päivän Scrumeihin, he kommunikoivat aidosti toistensa kanssa asioista, jotka oikeasti vaativat yhteistä käsittelyä. (Moe ym. 2009, 8.) Päivän Scrumien oikeanlaiseen toteutukseen tuleekin kiinnittää huomiota. Niiden tulee palvella koko tiimiä eikä vain toimia Scrum-mestarin ajan tasalla pysymisen välineinä.

Tutkimuksessa todettiin, että ilman selkeää ymmärrystä kehitettävästä järjestelmästä, suunnittelu oli vaikeaa. Kuukausittaisista suunnittelupalavereista muodostui enemmän Scrum-mestarin ja tuoteomistajan välistä keskustelua kuin aidosti kehitystiimin työtä tukevia tilaisuuksia. Tämä kommunikaation puute oli yksi syy siihen, että joskus tärkeitä tehtäviä ei tunnistettu ennen kuin vasta sprintin lopussa. Tämä vähensi yhteisen työlistan pätevyyttä, heikensi kommunikointia ja yhteistyötä entisestään, ja johti liian itsenäiseen työntekoon tiimin yhteistoiminnan kustannuksella. (Moe ym. 2009, 8.)

Tässä vaiheessa tiimin jäsenet kiinnittivät jo oikeaoppisesti joskus huomiota toistensa koodeihin. Palautteen ja muutosehdotusten antaminen ei kuitenkaan ollut aina helppoa. Erityisiä ongelmia oli silloin, jos tiimin jäsenet eivät luottaneet toisiinsa kunnolla tai eivät aidosti pitäneet toisistaan. Lisäksi tilanne tuntui hankalalta, jos palautteen antaja ei itsekään ollut täysin varma, oliko hänen tapansa parempi. (Moe ym. 2009, 8.)

Moen ym. tutkimuksessa viidennen ja kuudennen sprinttien kynnyskysymykseksi nousi aliurakoitsijan myöhässä toimittama ohjelmistomoduuli, joka ei toiminut toivotulla tavalla. Tämä tuli yllätyksenä tiimille ja vaati ongelmien korjaamista jopa ylitöinä. Moduulin integroiminen oli työlästä, ja johti siihen että työlistalla olevia asioita ei saatu tehtyä valmiiksi. Kehitystiimi sortui merkitsemään

keskeneräisiä ominaisuuksia valmiiksi, ja Scrum-mestarikin halusi näyttää ulospäin parempaa kuvaa kuin mitä todellisuus oli. Keskeneräisiä tehtäviä ei enää merkitty tulevien sprinttien työlistoille, mutta silti niihin jouduttiin käyttämään aikaa. Tämä johti siihen, että sprinttien aluksi piti tehdä valmiiksi merkittyjä, mutta oikeasti keskeneräisiä töitä, eikä kyseiselle sprintille merkittyjä ominaisuuksia saatu tehtyä. (2009, 8.)

Yrityksessä oli paineita uuden projektin alkamiseen, jolloin useiden ohjelmistokehittäjien olisi pitänyt siirtyä työskentelemään kokoaikaisesti uudessa projektissa. Keskeneräinen projekti menetti resursseja ja sen valmistuminen ei onnistunut alkuperäisessä aikataulussa. Heikosta varmistuksesta ja koodin itsenäisestä työstämisestä aiheutui ongelmia, eivätkä ohjelmoijat halunneet korjata toisten tekemiä ohjelmointivirheitä. Lisäksi tässä vaiheessa huomattiin, että joitakin kokonaisvaltaisia teknisiä ratkaisuja ei ollut osattu ottaa huomioon, koska suunnitelmia ei pohdittu yhdessä. Kun osa projektin avainhenkilöistä siirtyi muihin tehtäviin, lopputiimillä oli projektissa käytetystä itsenäisestä työskentelytavasta johtuen vaikeuksia saattaa projekti loppuun. (Moen ym. 2009, 8–9.)

4.2.2 Käyttöönottoon vaikuttavia tekijöitä

Kuten edellä kuvatun tutkimuksen aikana havaituista ongelmatilanteista voidaan päätellä, yksi tapa epäonnistua Scrumin käyttöönotossa on se, ettei organisaatiossa sitouduta aidosti Scrumin arvoihin ja käytäntöihin. Deemerin ym. (2010, 19) mukaan monet haasteista liittyvätkin juuri siihen, ettei Scrumin arvoja, käytäntöjä ja periaatteita ole sisäistetty kunnolla.

Scrumin perusarvot – sitoutuminen, keskittyminen, avoimuus, kunnioitus ja rohkeus – voivat olla ihan uusia monissa organisaatioissa. Sen vuoksi niiden toteuttaminen käytännössä ei välttämättä käy helposti. Osa työntekijöistä on tottunut siihen, että heille kerrotaan mitä heidän tulee tehdä ja miten. Työntekijät voivat olla epäluuloisia, eivätkä välttämättä heti ymmärrä, että johto antaa heidän toimia Scrumin arvojen mukaisesti. (Schwaber & Beedle 2001, 148.) Näin ollen yrityksen johdon ja erityisesti Scrum-mestarin tulee kannustaa työntekijöitä arvojen toteuttamiseen ja kertoa heille selkeästi, mitä heiltä odotetaan. Kun työntekijät kokevat olevansa täysivaltaisia tiimin jäseniä, he alkavat vähitellen uskoa itseensä (Schwaber & Beedle 2001, 148). Tällöin kehitystiimin luottamus oman työnsä jälkeen kasvaa ja projektin lopputuloksena olevasta ohjelmistosta voidaan odottaa laadultaan hyvää.

On kuitenkin huomioitava, että aina työntekijät eivät ole valmiita sitoutumaan Scrumin periaatteisiin, vaikka heillä olisikin johto tukenaan. Sama voi koskea myös yrityksen johtoa ja esimiehiä. Schwaberin mukaan jopa 20 prosenttia ohjelmistokehittäjistä ja johtajista voi päätyä lähtemään yrityksestä. Osa työntekijöistä ei halua muuttaa tapaansa tehdä vain ne työt, jotka heille annetaan. Johtajat voivat olla haluttomia muuttamaan oppimiaan johtamistapoja. Vaatimus muunlaisen sitoutumiseen voi aiheuttaa heissä tyytymättömyyttä. (2007, 5–7.)

Jotta Scrumin käyttöönotossa voidaan onnistua, on yrityksestä löydyttävä motivoituneita ja sitoutuneita työntekijöitä viemään projektia eteenpäin. Schwaberin mukaan asiansa osaava, tehtäviinsä paneutunut ja niihin riittävästi aikaa käyttävä Scrum-mestari on avainasemassa Scrumin käyttöönoton onnistumisessa. Toisille Scrum-mestarin tehtävien oppiminen on helppoa, toisilla on vaikeuksia ymmärtää tehtävän vaatimuksia ja toiset eivät asiaan perehtymisestään huolimatta edes kykene toimimaan tässä roolissa. Sekin täytyy muistaa, että myös tehtävän jossain määrin sisäistänyt Scrum-mestari tekee välillä harmittavia virheitä. Tarvitaankin kokemuksia useista sprinteistä, ennen kuin Scrum-mestarin rooli sujuu luontevasti. (2004, 25, 36.)

Scrum-mestarin täytyy sisäistää Scrumin taustalla oleva filosofia. Scrumia ei pidä tulkita suhteessa aikaisemmin käytössä olleeseen ohjelmistokehitysmenetelmään. Scrum-mestarin tulee ymmärtää, että Scrumiin kuuluu siirtyminen kontrolloinnista mahdollistamiseen, sopimuksista yhteistyöhön ja dokumentaatiosta koodiin. Hänen täytyy selviytyä uuden roolinsa mukanaan tuomista muutoksista ja asennoitua uuteen johtamistapaan. (Schwaber 2004, 25–26, 105.)

Monet perinteisesti projektipäällikölle kuuluneista tehtävistä ovat Scrumia käytettäessä kehitystiimin vastuulle kuuluvia asioita (Virtanen 17.6.2010, luento). Scrum-mestarin täytyy ymmärtää – ja huolehtia, että myös kehitystiimi ymmärtää – ettei hänen pidä tehdä sitoumuksia tiimin puolesta. Tiimin itsensä tulee määritellä, mitä ominaisuuksia ja kuinka paljon tietyssä ajassa voidaan toteuttaa. Kehitystiimin tulee pitää huolta omasta edistymisestään, että aikataulussa pysytään ja että ongelmiin puututaan. Tiimin tulee määritellä itse ratkaisut, mikäli aikataulutus ei olekaan onnistunut tai muita ongelmia ilmenee. Scrum-mestarin ei pidä pakottaa tiimiläisiä työskentelemään lujemmin, mikäli he eivät itse koe siihen pystyvänsä. Tätä ei pidä tehdä edes lupaamalla ylimääräisiä palkkioita. Scrum-mestarin ei tule määritellä mitä kunkin tiimiläisen tulee tehdä seuraavaksi. Hänen ei pidä valvoa tehtävien toteutumista, vaan tiimin itsensä pitää huolehtia siitä, että jokainen tehtävä tulee tehtyä valmiiksi. (Tieturi Oy 2010, 70.)

Kehitystiimin itseohjautumisen sisäistäminen on asia, johon sekä Scrum-mestarin että tiimin itsensä tulee kiinnittää huomiota. Tiimin tulee ymmärtää, että se on vastuussa omista tuloksistaan ja että se saa päättää itsenäisesti, millaisia keinoja päämääriin pääsemiseksi käytetään. Tämän mahdollistamiseksi tiimistä pitää löytyä kaikkea sitä osaamista, jota tehtävien suorittaminen edellyttää. Scrum-mestarin tuleekin huolehtia tiimin kokoonpanosta ja tietotaidosta. (Tieturi Oy 2010, 64.)

Mikäli Scrum on kehitystiimille vielä uusi asia, kokeneistakin IT-alan ammattilaisista muodostuvalla tiimillä saattaa olla ensimmäisten sprinttien aikana ongelmia. Tiimi ei välttämättä vielä ymmärrä, että sen tulee johtaa itse itseään ja huolehtia suunnittelusta itsenäisesti. Tiimiläiset saattavat odottaa, että sprintin suunnittelukokouksessa heille kerrotaan, mitä heidän tulee tehdä. Siirtyminen projektipäällikön johtamasta tiimistä itseohjautuvaksi kehitystiimiksi ei olekaan helppoa. Itseohjautuminen on aloittelevalle tiimille vain abstrakti termi, josta ei ole vielä todellista käsitystä. Tiimi tarvitsee konkreettista kokemusta Scrumista ennen kuin se voi kunnolla ymmärtää, miten omaa toimintaa ohjataan ja miten otetaan vastuu suunnittelusta ja omasta aktiivisuudesta. (Schwaber 2004, 101–102, 104–105.)

Scrum-mestarin tehtävänä on valmentaa tiimiä tässä työtavan muutoksessa. Hän voi esimerkiksi kertoa, että sprintin katselmointikokoukseen on aikaa 29 päivää ja ettei kukaan tule kertomaan kehitystiimiläisille, mitä heidän pitää tehdä, vaan heidän pitää selvittää tarvittavat toimenpiteet itse. Oletettavasti pienen hetken kuluttua joku tiimin jäsenistä ehdottaa ensimmäistä toimenpidettä ja aloittaa siten kehitystiimin muuttumisen päällikkövetoisesta tiimistä itseohjautuvaksi tiimiksi. (Schwaber 2004, 101–102.)

Schwaberin mukaan itseohjautuva kehitystiimi ei siis tarkoita sitä, että Scrum-mestarin tiivistä panosta ei tarvittaisi projektien aikana. Schwaber vertaa Scrum-mestaria lammaslaumaa paimentavaan paimenkoiraan, joka ei kesken paimentamisen laita nukkumaan, vaan on aina valppaana ohjaamaan ja puolustamaan laumaansa. Tämä vertauskuva voidaan käsittää siten, että Scrum-mestarin pitää olla aina valmiina tiimiä varten ja ehdottoman motivoitunut hoitamaan rooliaan. (2004, 30.) Myös Virtanen nostaa esille sen, että Scrum-mestarin motivaatio tehtäviinsä on ensiarvoisen tärkeää. Kun hän on kiinnostunut ja innostunut, hänen on helpompi innostaa ja motivoida kehitystiimiäkin. (17.6.2010, luento.) Schwaberin mukaan kehitystiimillä on oltava tunne siitä, että Scrum-mestari panostaa tiimin tukemiseen ja auttaa ja turvaa tiimiä toiminnassaan. Hänen ei pidä antaa tiimille sellaista viestiä, että heidän työnsä on hänen mielestään

vähäarvoista. Esimerkiksi tilanne, jossa Scrum-mestari pitää jotain ulkopuolista palaveria tärkeämpänä kuin tiimin kanssa jo sovittua palaveria, voi aiheuttaa tiimin jäsenissä epäluottamusta. (2004, 30.)

On kyseessä sitten kokeneista ammattilaisista koostuva tiimi tai tiimi, jossa on myös kokemattomampia jäseniä, tiimin toimivuus on olennaisen tärkeää. Toimimattomassa tiimissä voidaan nähdä erilaisia ongelmia, jotka haittaavat projektin etenemistä. Tieturi Oy:n Certified ScrumMaster -materiaalissa viitataan Patric Lencionin määrittelemiin toimimattoman tiimin piirteisiin. Useimmat toimimattoman tiimin ongelmista pohjaavat siihen epäkohtaan, että tiimin jäsenet eivät luota toisiinsa. Luottamuksen puutteen myötä tiimissä voi esiintyä myös sitoutumisen puutetta ja konfliktien pelkoa. Tästä aiheutuu vastuun välttelyä eikä projektin lopputulokseen tule kiinnitettyä riittävästi huomiota. (2010, 69.) Virtasen mukaan toimivan tiimin pitää siis luottaa toisiinsa. Tiimin jäsenten tulee uskaltaa olla rohkeasti eri mieltä, mikäli tilanne niin vaatii. Heidän tulee olla sitoutuneita ja pitää huolta siitä, että kukin tiimiläinen voi hoitaa tehtävänsä. Tiimin tulee muistaa pyrkiä hyvään lopputulokseen ja korostaa onnistumisen merkitystä. (17.6.2010, luento.)

Toimiva tiimi ei tarkoita sitä, että kaikki tiimin jäsenet ovat samaa mieltä asioista. Toimivassakin tiimissä voidaan – ja pitää – olla eri mieltä, kun tilanne niin vaatii. Schwaberin mukaan erilaisten mielipiteiden ja kantojen esiin nostaminen on vain hyvä asia, sillä konflikti on merkki muutoksesta ja pyrkimyksestä asioiden parantamiseen. Hän korostaakin, että tiimin sisäisiä konflikteja on odotettavissa eikä niitä pidä hätkähtää. Ihmiset eivät kuitenkaan välttämättä osaa käsitellä ristiriitatilanteita rakentavasti. Sen vuoksi Scrum-mestarin tulee kiinnittää huomiota asiaan ja kouluttaa tiimiläisiä konfliktien ratkaisemisessa. (2007, 6.)

Scrum-mestarin tulee muistaa kannustaa tiimiläisiä ja tukea heitä heidän tehdessään itsenäisiä päätöksiä tehtäviensä suhteen (Schwaber & Beedle 2001, 152). Hänen täytyy kuitenkin huomioida kehitystiimin jäsenten opastamisessa näiden kokemus- ja osaamistaso. Mikäli ohjelmistokehittäjällä ei ole vielä riittävästi osaamista ja rohkeutta, tulee hänen suhteensa käyttää perinteistä johtamistapaa. Kun työntekijältä alkaa löytyä osaamista, intoa ja rohkeutta, Scrum-mestari voi kannustaa häntä ottamaan lisää vastuuta. Vasta kun työntekijä suoriutuu hänelle annetuista tehtävistä itsenäisesti ja rohkeasti, hän voi toimia täysipainoisesti itseohjautuvan tiimin jäsenenä. (Virtanen 17.6.2010, luento.)

Scrumin arvojen mukaisesti kehitystiimiltä ja sen jäseniltä edellytetään rohkeutta. Tämä ei ole välttämättä helppoa, sillä tiimin jäsenet voivat arastella mielipiteidensä kertomista ja avuntarpeensa esilletuomista. He voivat syyttää itseään siitä, etteivät ymmärrä jotain tai etteivät ole riittävän hyviä. Jokaisen jäsenen tulee kuitenkin muistaa, että kullakin on omanlaisensa tausta, koulutus ja työkokemus. Tavoitteena ei ole se, että osaa kaiken kaikissa tilanteissa vaan että osaa myöntää omat heikkoutensa ja vahvuutensa. Että osaa pyytää rohkeasti apua sitä tarvitessaan – ja osaa tarjota apua toisen sitä kaivatessa. (Schwaber & Beedle 2001, 152, 154.)

Ihannetilanteessa tiimiin valitut yksilöt muodostavat toisiaan kunnioittavan, itseohjautuvan tiimin. Tämä ei läheskään aina suju ongelmitta. Henkilökemioiden ohella monenkirjavat teknologiat ja muuttuvat vaatimukset voivat aiheuttaa hankaluuksia. Voi käydä esimerkiksi niin, että joku työntekijöistä ei pysty suoriutumaan tehtävistään niin hyvin kuin tiimin tehokkuuden kannalta olisi tarpeellista. Hän ei esimerkiksi ole aikaisemmin toteuttanut ratkaisuja projektissa käytettävillä tekniikoilla. Tällöin toiset työntekijät voivat olla pettyneitä hänen toimintaansa ja kohdella häntä epäkunnioittavasti. Scrumin periaatteiden mukaisesti tiimin ei pitäisi kuitenkaan keskittyä siihen, mitä jää tekemättä, vaan siihen, miten tilannetta voitaisiin parantaa. Tiimiläisten tulisi yhdessä miettiä ratkaisukeinoja tilanteeseen ja tukea hankaluuksissa olevaa työtoveriaan. Yksi ratkaisu on, että tilanteessa käytetään ulkopuolista asiantuntijaa, joka opastaa työntekijää ja auttaa häntä ymmärtämään uuden teknologian käyttöä. Tiimin jäsenten on tärkeää muistaa, että jokainen yrittää tehdä parhaansa ja että työtoveria tulee auttaa, mikäli vain suinkin voi. (Schwaber & Beedle 2001, 152–153.)

Scrum ei sinänsä määrittele, mitä ohjelmointi- ja testausmenetelmiä tiimin tulisi kehitystyössään käyttää. Mutta näihinkin asioihin on syytä kiinnittää huomiota ja sopia yhteisesti noudatettavista käytännöistä. Hyvät ohjelmointi- ja testauskäytännöt ovat edellytyksenä sille, että tiimi voi toimittaa toimivia ohjelmiston osia. Myös Päivän Scrumien tehokkuuden kannalta on tärkeää, että tiimi todella tietää, missä vaiheessa työt ovat ja mitä täytyy vielä tehdä, ennen kuin jokin osa-alue voidaan hyväksyä valmiiksi. Mikäli käytännöt eivät ole kunnossa, Scrum-mestarin tulee opastaa kehitystiimiä löytämään hyvät ohjelmointi- ja testausmenetelmät. Tähän tulee käyttää riittävästi aikaa, tarvittaessa useita viikkoja. Voidaan päättää hyödyntää esimerkiksi Extreme Programming -menetelmän käytäntöjä. (Schwaber 2004, 106–107.)

Jatkuva keskittyminen työtehtävänä olevan ongelman ratkaisemiseen on uuvuttavaa työtä. Siitäkin huolimatta työntekijöiden pitäisi keskittyä yhteen tehtävään kerrallaan ja unohtaa kaikki

asiaankuulumaton. Asioiden edelle meneminen ja vasta myöhemmässä vaiheessa ajankohtaisiksi tulevien tehtävien pohdiskelu etukäteen voi harhauttaa kehitystiimiläisiä ja viedä turhaan aikaa työn alla olevilta tehtäviltä. Tällaisessa tilanteessa Scrum-mestarin tuleekin esittää Päivän Scrumissa ohjelmistoprojekteihin osallistuville työntekijöille kysymys "Mitä asialla on tekemistä koodin kanssa?" Tämä auttaa kehitystiimiläisiä keskittymään tehokkaan työskentelyn kannalta oleellisiin asioihin. (Schwaber & Beedle 2001, 149–150.)

Scrum-mestarin tulee tiedostaa oma roolinsa fasilitaattorina ja valmentajana kontrolloimisen ja johtamisen sijaan. Jotta Päivän Scrum (katso luku 3.4.4) voi toimia oikealla tavalla, Scrum-mestarin tulee ymmärtää oikealla tavalla siinä käsiteltävät kysymykset. Hänen ei pidä peilata niitä mahdolliseen aikaisempaan kokemukseensa perinteisestä projektin johtamisesta. Muuten seurauksena voi olla Scrumin periaatteisiin sopimaton tilanne, jossa Scrum-mestari tarkistaa Päivän Scrumissa ovatko kehitystiimin jäsenet tehneet ne toimet, jotka hän ohjeisti heidät tekemään edellisessä Päivän Scrumissa. Lisäksi hän voi päätyä kertomaan jokaiselle tiimiläiselle, mitä heidän tulisi tehdä ennen seuraavaa päivää. Hän voi myös tulkita kysymyksen tiimin kohtaamista esteistä siten, että vain yksipuolisesti tarkistaa, voisiko hän tehdä jotain auttaakseen tiimiä pääsemään tavoitteisiin. Kehitystiimin kyky torjua ja ratkaista ongelmat itsenäisesti on Scrumin ydin ja Scrum-tiimin tuottavuuden perusta. Tämän vuoksi Scrum-mestarin tulee aidosti tukea tiimiä myös Päivän Scrumeissa eikä sortua autoritaariseen johtamiseen. (Schwaber 2004, 28–29.)

5 SCRUMIN KÄYTTÖÖNOTTOSELVITYS TOIMEKSIANTAJAYRITYKSELLE

Toimeksiantajayrityksen lähtökohtatilannetta, toimintatapoja ja projektinhallinnan kehittämiselle esitettäviä toiveita ja odotuksia selvitettiin haastattelemalla kahta yrityksen edustajaa, joita käsitellään työssä anonyymisti. Haastattelua varten laadittiin tukikysymyksiä, joissa käsitellään yrityksen ja sen projektinhallinnan lähtökohtatilannetta, nykyistä projekti- ja tiimityöskentelyä sekä Scrumia (liite 5). Osa kysymyksistä lähetettiin haastateltaville ennen haastattelua tutustumista varten.

Toinen haastatelluista toimii toimeksiantajayrityksessä yrittäjänä ja projektipäällikkönä. Hän huolehtii projektien vetämisestä ja yhteydenpidosta asiakkaisiin. Käytännössä hänen vastuullaan on asiakkaan esittämien vaatimusten kartoittaminen ja dokumentointi, ja tämän tiedon välittäminen muille projektitiimin jäsenille. Projektin vetäjänä hän huolehtii siitä, että asiat tulevat tehdyiksi. Hän myös jakaa tarvittaessa tehtäviä, organisoi ja johtaa. (Projektipäällikkö 15.4.2011, haastattelu.) Toinen haastatelluista työskentelee yrityksessä kokopäiväisenä ohjelmistosuunnittelijana. Hänen tehtävänsä käsittävät pääasiassa web-ohjelmointia, ohjelmiston ja tietokannan suunnittelua sekä käyttöliittymän rakentamista. (Ohjelmistosuunnittelija 18.4.2011, haastattelu.) Toimeksiantajan nykytilanteen arvioinnissa ja toiminnan kehittämisessä on hyödynnetty myös opinnäytetyön tekijän omakohtaisia havaintoja, joita hän kirjasi ylös työskennellessään toimeksiantajayrityksessä kahteen otteeseen vuosina 2009–2010.

5.1 Toimeksiantajan lähtökohtatilanne

Toimeksiantajayrityksen palveluksessa on tilanteesta riippuen kolme–viisi työntekijää, joista osa on kokoaikaisia ja loput osa-aikaisia. Yrityksessä työskentelee välillä myös opiskelijoita, sillä yritys on tarjonnut työtä harjoittelupaikkojen muodossa ammattikorkeakoulu- ja yliopisto-opiskelijoille sekä valmistumisensa jälkeen esimerkiksi työvoimapoliittisessa koulutuksessa oleville henkilöille. Osa näistä opiskelijoista on harjoittelujaksonsa päätyttyä työllistynyt yritykseen. Haastatteluhetkellä yrityksessä oli yksi harjoittelija. (Projektipäällikkö 15.4.2011, haastattelu.)

Yrityksessä on käynnissä samanaikaisesti useita ohjelmistoprojekteja. Projektipäällikkö kertoi, että haastatteluhetkellä käynnissä on kaksi pääprojektia ja pienempi tukiprojekti, jonka parissa työskennellään tarvittaessa. Lisäksi yrityksellä on useampia projekteja, joiden kehitystyö ei ole

vielä päättynyt, ja näiden projektien pariin täytyy ajoittain palata. Projektipäällikön mukaan tyypillisessä ohjelmistokehitysprojektissa työskentelee hänet lisäkseen yhdestä kolmeen henkilöä. Tulevaisuudessa projektiin osallistuvien työntekijöiden määrä voi lisääntyä yrityksen kasvun myötä. (15.4.2011, haastattelu.)

Sama työntekijä voi työskennellä yhtä aikaa useassa projektissa. Usein tilanne on se, että joku projekteista tai sen ominaisuuksista on odottamassa jotain tietoa tai jonkin toisen ominaisuuden valmistumista. Tällä välin työntekijä voi työstää toisen projektin asioita. (Projektipäällikkö 15.4.2011, haastattelu.) Tällainen tilanne, jossa ei tule toimettomia aikoja, on työntekijöiden työllistymisen ja kustannustehokkuuden kannalta hyvä. Projektipäällikkö totesi, että projektitiimin jäsenet joutuvat osallistumaan myös ohjelmistojen tuki- ja huoltotehtäviin, usein hyvin yllättäen ja lyhyellä varoitusajalla. Tällainen äkillinen siirtyminen tehtävästä toiseen rikkoo ja häiritsee työskentelyä ja aiheuttaa enemmän virheitä lopputulokseen. Työntekijät ovatkin kokeneet haasteellisena tällaisen äkillisen, jopa saman päivän aikana tapahtuvan projektista toiseen siirtymisen. Tähän on yrityksessä jo puututtu siten, että samana päivänä ei siirrytä esimerkiksi pääprojektista tukiprojektiin, vaan toisen projektin parissa työskentely aloitetaan vasta seuraavana aamuna. (15.4.2011, haastattelu.)

Pienen IT-yrityksen toiminnan kannalta haasteellista on, että jokaisen työntekijän osaamisalueen täytyy olla melko laaja. Toisaalta tämä lisää työn mielekkyyttä, kun ohjelmistokehittäjät saavat tehdä monipuolisia tehtäviä. Toimeksiantajayrityksen projektitiimin jäseniltä löytyy tarvittavaa osaamista niin ohjelmistosuunnittelusta, graafisesta suunnittelusta kuin ohjelmoinnistakin (Ohjelmistosuunnittelija 18.4.2011, haastattelu). Haastatteluissa tuli esille se, että tiimiläisillä ei ole koulutusta projektin johtamisesta eikä testauksesta, vaan nämä osa-alueet ovat itse opittuja. Tämän ei ole koettu kuitenkaan aiheuttaneen ongelmia. (Projektipäällikkö 15.4.2011, haastattelu; Ohjelmistosuunnittelija 18.4.2011, haastattelu.) Ohjelmistosuunnittelija (18.4.2011, haastattelu) toi esille sen, että jos on tullut tarvetta jollekin uudelle tiedolle tai taidolle, niin aiheesta on otettu selvää ja opeteltu työn ohessa.

Koska toimeksiantajayrityksen projektitiimit ovat pieniä eikä testauskäytäntöjä ole kehitetty loppuun asti, jokainen tiimin jäsen joutuu testaamaan myös omia tuotoksiaan. Tällöin joitakin asioita voi jäädä huomaamatta. Tiimissä on ratkottu tätä ongelmaa siten, että testataan ristiin toisten toteuttamia ominaisuuksia tai ohjelmiston osia mahdollisuuksien mukaan. Ristiin testattaessa on olemassa vaara, että sorrutaan liian tekniseen, insinöörimäiseen näkökulmaan, koska

testaajat ovat samalla itsekin ohjelmoijia. Tämä tuo haasteita varsinkin käytettävyystestaukseen, koska sitä tehdessä pitää osata ajatella, miten ohjelmiston loppukäyttäjä ohjelmistoa käyttää. (Ohjelmistosuunnittelija 18.4.2011, haastattelu.)

Projektipäällikkö toi esille sen, että pienessä tiimissä ohjelmistokehitysprosessia ei välttämättä pidetä kovin tärkeänä työn tekemisen kannalta. Hän on saanut sen vaikutelman, että työntekijät ajattelevat töiden hoituvan omalla painollaan ilman huomion kiinnittämistä prosessiin. (15.4.2011, haastattelu.)

Ohjelmistosuunnittelijan mukaan tiimin jäsenillä on aika hyvä käsitys siitä, mitä kukin milloinkin tekee. Tähän vaikuttaa se, että kaikki ohjelmistokehittäjät työskentelevät samassa huoneessa. (18.4.2011, haastattelu.) Lisäksi projektipäällikkö kertoi yrityksessä käytettävästä projektinhallinnan ohjelmistosta, jonka kautta sekä projektin vetäjä että projektitiimin jäsenet voivat seurata tehtävien etenemistä. Ohjelmiston käytöstä on tehty työntekijöille ohjeistus, jonka mukaisesti merkintöjä tehdään yksittäisen työtehtävän aloituksesta, etenemisestä ja päättymisestä. (15.4.2011, haastattelu.) Ohjelmistosuunnittelija näki projektinhallinnan ohjelmiston käytöstä olevan hyötyä, mutta koki yhteisten toimintamallien sopimisessa parannettavaa. Hänen mielestään ominaisuuksia toteutettaessa tulisi edetä saman mallin mukaisesti tiettyjä työvaiheita noudattaen, mikä parantaisi tiimin tehokkuutta ja työn laatua. (18.4.2011, haastattelu.)

Projektipäällikön mukaan projekteja aikataulutetaan tällä hetkellä vain sen valmistumispäivän mukaan. Sille, milloin kunkin ohjelmiston osa-alueen tai ominaisuuden tulee olla valmiina, ei ole asetettu päivämääriä. Yrityksessä ei siis seurata työmäärän vähenemistä, mutta työn tulee silti olla valmis tiettynä päivänä. (15.4.2011, haastattelu.)

Sekä ohjelmistosuunnittelija että projektipäällikkö kertoivat, että yrityksessä ollaan ottamassa käyttöön uutta versionhallintaohjelmistoa. Sen uskotaan muuttavan työskentelytapoja tehokkaammaksi. Haastatteluvastauksista ei kuitenkaan käynyt ilmi, miten tehostamista odotetaan tapahtuvan. (18.4.2011, haastattelu; 15.4.2011, haastattelu.)

Osa-aikaisten työntekijöiden käyttäminen aiheuttaa sitä, että tehtävien koetaan olevan hajallaan (Projektipäällikkö 15.4.2011, haastattelu). Osa-aikaiset työntekijät voivat aloittaa jonkin työtehtävän, mutta mikäli se ei valmistukaan suunnitellussa aikataulussa, voi se jäädä muiden työntekijöiden vastuulle. Ohjelmistosuunnittelija ei kuitenkaan ole kokenut toisten aloittaman työn

jatkamista ongelmallisena, sillä yrityksen käyttämä ohjelmistoalusta pakottaa tietynlaiseen ohjelmointiin ja ohjelmistokoodin kommentoinnista on sovittu yhteisiä pelisääntöjä. Lisäksi tarvittaessa on ollut mahdollisuus kysyä lisätietoja henkilöltä, joka on aloittanut kyseisen ominaisuuden toteuttamisen. Yksittäiset työntekijät eivät "omista" tekemäänsä koodia, vaan kaikilla on oikeus tehdä siihen muutoksia, lisäyksiä, parannuksia ja korjauksia. (18.4.2011, haastattelu.)

Toimeksiantajayrityksessä suhtaudutaan joustavasti projektin toteutuksen aikana muuttuviin vaatimuksiin. Projektipäällikön mukaan huolellisesta vaatimusmäärittelyn laatimisesta huolimatta projektin edetessä saatetaan huomata, että jonkin osa-alueen toteuttaminen vaatii vielä lisätarkennuksia. Aikaisemmin tällaiset tarkennukset saattoivat vaikuttaa projektin etenemiseen ja kokonaisuuteen paljonkin, mutta nyt määrittelyn ei anneta enää rikkoa projektia. Lähtökohtana on, että asiakkaan kanssa tehtävä vaatimusmäärittely tehdään niin tarkaksi, ettei sitä ole tarvetta muuttaa myöhemmässä vaiheessa, ja asiakas sitoutuu sovittuihin asioihin hyväksymällä määrittelydokumentin. (15.4.2011, haastattelu.) Ohjelmistosuunnittelija kertoi, että asiakkaalta voi tulla projektin aikana muutosehdotuksia tai -pyyntöjä. Tällöin projektipäällikön tehtävänä on arvioida, mitkä voidaan toteuttaa sovitun vaatimusmäärittelyn puitteissa. Mikäli asiakkaalla ei ole aluksi selkeää kuvaa siitä, mitä lopputulokselta halutaan, projektitiimi työstää ohjelmiston osa-alueita projektin edetessä syntyneiden ideoidensa pohjalta. Isommat kehitysideat kirjataan kuitenkin vain ylös, olivat ne sitten lähtöisin asiakkaalta tai projektitiimiltä itseltään. Näihin ideoihin palataan myöhemmässä vaiheessa, jolloin päätetään toteutetaanko niitä. Kustannustehokkaiksi ja teknisiltä ominaisuuksiltaan hyödyllisiksi todetut ominaisuudet toteutetaan omana projektinaan. (18.4.2011, haastattelu.)

Projektin tavoite asetetaan projektipäällikön (15.4.2011, haastattelu) mielestä lähinnä aikataulullisesti. Ohjelmistosuunnittelija kokee, että asiakkaan kanssa laadittu vaatimusmäärittely konkretisoi tavoitetta: projektissa tehdään sille varatussa ajassa mahdollisimman kannattavasti se, mitä määrittelydokumenttiin on kirjoitettu. Tavoitteena on mahdollisimman toimiva lopputulos, johon asiakas on tyytyväinen. Nämä seikat huomioiden ohjelmistosuunnittelija totesi, että projektin tavoitteet ovat olleet aika selkeitä. (18.4.2011, haastattelu.)

Projektitiimin jäsenet työskentelevät samassa huoneessa, mikä helpottaa konkreettisesti tiedon jakamista ja kommunikaatiota. Tiimiläisten on helppo keskustella, jakaa osaamistaan ja pyytää tarvittaessa apua tai mielipidettä. (Ohjelmistosuunnittelija 18.4.2011, haastattelu.) Projektipäällikkö näkee kuitenkin tarvetta keskinäisen tiedonvälityksen parantamiseen. Hän kertoi esi-

merkkinä, että tiimin jäsenet saattavat toteutusvaiheessa yllättyä toisten tekemistä ratkaisuista, vaikka he ovat lukeneet saman määrittelyn ja jakaneet tehtävät yhdessä. Projektipäällikön mielestä esimerkiksi dokumentaation kehittäminen tai kunkin tiiminjäsenen toteuttamien ominaisuuksien läpikäyminen yhdessä voisi auttaa tiedonvälitystä. (15.4.2011, haastattelu.)

Projektipäällikkö kertoi, että palautteen antamiseen ei ole olemassa mitään tiettyä menetelmää tai yhdessä pohdittua tapaa. Hän itse antaa tarvittaessa palautetta erilaisista asioista ja huomauttaa esimerkiksi, jos yrityksessä käyttöönotettuja menetelmiä tai ohjeistuksia ei ole noudatettu riittävästi. Hänen mukaansa kehitystiimin jäsenet antavat toisilleen palautetta ainakin jossain määrin. (15.4.2011, haastattelu.) Ohjelmistosuunnittelijan mielestä kehitystiimin sisällä annetaan riittävästi palautetta, ja sitä osataan myös kysyä työkaverilta. Näin toimitaan esimerkiksi silloin, kun työn alla on jokin uutta ratkaisua vaativa asia, johon kaivataan kollegan neuvoa tai jota toivotaan pohdittavan yhdessä. Tiimissä tunnistetaan myös toisten vahvuudet ja osaamisalueet. Ohjelmistosuunnittelija kertoi, että esimerkiksi ulkoasua toteuttaessaan hän kysyy usein mielipidettä tiimikaveriltaan, jolla on enemmän graafisen alan koulututusta ja tietämystä. (18.4.2011, haastattelu.)

Mikäli tiimikaverin käyttämässä ohjelmointitavassa huomataan jokin virhe tai muu parantamista vaativa seikka, siitä huomautetaan suoraan (Projektipäällikkö 15.4.2011). Myös ohjelmistosuunnittelija toi esille, että toisen tekemään koodiin liittyen annetaan suoraa palautetta. Esimerkiksi, jos toisen tekemään osaan muutoksia tehtäessä keksitään parempi tai toimivampi tapa tehdä kyseinen asia. Uusista huomioista kerrotaan tiimikavereille myös silloin, kun tehdään parannuksia itse tehtyyn koodiin. Tämä tieto halutaan välittää koko tiimille, jotta vastaavissa tilanteissa muutkin voivat hyödyntää tätä uutta osaamista. (18.4.2011, haastattelu.)

Toimeksiantajayrityksessä on kiinnostuttu Scrumista, koska sen koetaan olevan menetelmä, jonka avulla voidaan saavuttaa nopeasti tuloksia. Scrumin käytäntöjen koetaan ohjaavan työntekoa siten, että kehitettävästä ohjelmistosta on jokaisen kehitysjakson päätteeksi jotain uutta esiteltävää asiakkaalle. Tällöin asiakasyrityksessä voidaan reagoida nopeasti siihen, vastaako toteutettu ohjelmiston osa heidän toiveitaan. (Projektipäällikkö 15.4.2011, haastattelu.)

Toimeksiantajayrityksessä on suunniteltu koko henkilöstön kouluttamista Scrumiin liittyen, mutta koulutuksen perusteellisuutta ja syvyyttä ei ole vielä päätetty. Osalla henkilöstöstä on Scrumista jo jonkinlainen näkemys, sillä yrittäjät ja yksi projektitiimin jäsenistä, haastateltu ohjelmistosuunnittelija, ovat osallistuneet kahden päivän mittaiseen Scrum-koulutukseen. (Projektipäällikkö

15.4.2011, haastattelu.) Ohjelmistosuunnittelijalla (18.4.2011, haastattelu) on mielestään hyvä pohjatieto Scrumista, mutta mikäli menetelmä otetaan yrityksessä käyttöön, hän kokee tarvitsevansa asioiden kertaamista.

Yrityksessä on vähitellen siirrytty projektimenetelmiin, jotka noudattelevat Scrumin periaatteita. Aamuisin pidetään enintään 15 minuutin mittainen palaveri, jossa kukin projektitiimin jäsen kertoo vuorollaan projektipäällikölle ja muille tiimin jäsenille Päivän Scrumin tapaan, mitä on tehnyt edellisenä päivänä, mitä aikoo tehdä tänään, ja onko ilmennyt joitain ongelmia (Projektipäällikkö 15.4.2011, haastattelu; Ohjelmistosuunnittelija 18.4.2011, haastattelu).

Aamupalavereiden lyhyestä kestosta huolehditaan. Työn tekemisessä kohdatut ongelmat nostetaan esille, mutta niitä käsitellään tarkemmin vasta palaverin jälkeen asianosaisten kesken. Myös siitä huolehditaan, että palaverit pidetään joka päivä. Yrityksessä on käytäntö, jonka mukaan kello 9.15 sovituksi alkavasta aamupalaverista myöhästynyt henkilö joutuu maksamaan sakkoa, ellei hän ole etukäteen ilmoittanut poissaolostaan. Sakon määrää ei kylläkään ole määritelty, mutta sen avulla kaikki tiedostavat, että on kyse tärkeästä asiasta. Aamupalaveri useimmiten etenee projektipäällikön johdolla, mutta jos hän ei pääse paikalle, palaveria vetää toinen henkilö. (Projektipäällikkö 15.4.2011, haastattelu.)

Yrityksessä oli kuitenkin joidenkin kuukausien ajan vaihe, jolloin ohjelmistoja kehitti vain yksi henkilö, ja silloin aamupalaverille ei nähty tarvetta. Kahden tai useamman työntekijän tiimissä siitä koetaan olevan hyötyä ja mitä enemmän projektitiimissä on jäseniä, sitä hyödyllisemmäksi palaveri koetaan. Vaikka projektitiimi työskenteleekin samassa työhuoneessa, yhteinen aamupalaveri selventää projektin tilannetta kaikille tiimin jäsenille. (Projektipäällikkö 15.4.2011, haastattelu.) Ohjelmistosuunnittelijankin mielestä päivittäiset palaverit ovat hyödyllisiä. Ensinnäkin hän saa hyvän käsityksen siitä, mitä muut tiimin jäsenet tekevät. Toisekseen hänen omat tehtävänsä selkiytyvät, kun hän sanoo ne kerran päivässä ääneen. Lisäksi hän uskoo, että projektipäällikön on helpompi aikatauluttaa asioita, kun tämä tietää, missä vaiheessa kukin työntekijä on menossa. (18.4.2011, haastattelu.)

Yrityksessä hyödynnetään jossain määrin Scrumin käytäntöjä myös julkaisun ja sprintin suunnittelupalaverin osalta. Jokaisen projektin alussa pidetään päivän mittainen palaveri, jossa tutustutaan vaatimusmäärittelyyn ja jaetaan toteutettava ohjelmisto työtehtäviin. Jokaisen sprintin alussa valitaan ne tehtävät, jotka sen aikana aiotaan toteuttaa. Osa näistä tehtävistä jaetaan jo

tässä vaiheessa tiimin jäsenille ja osa jätetään sprintin aikana poimittaviksi. (Projektipäällikkö 15.4.2011, haastattelu.)

Työtehtävät pyritään rajaamaan niin pieniksi, että niiden toteuttamiseen kuluu enintään yksi päivä (Ohjelmistosuunnittelija 18.4.2011, haastattelu). Tässä on vielä kehitettävää, mutta rajaamista ja tehtäviin tarvittavan ajan arvioimista on opittu koko ajan enemmän projektitiimin asiantuntemuksen ja osaamisen kasvaessa (Projektipäällikkö 15.4.2011, haastattelu; Ohjelmistosuunnittelija 18.4.2011, haastattelu). Projektipäällikön (15.4.2011, haastattelu) mukaan pitempään yrityksessä työskennelleet työntekijät tukevat ja opastavat kokemattomampia työntekijöitä niin tehtävien työmäärän arvioimisessa kuin muissakin yrityksen käyttämissä ohjelmistokehitysmenetelmissä.

Projektipäällikkö kertoi, että yrityksessä järjestetään kunkin sprintin päätteeksi kehityskeskustelu, joka noudattelee jossain määrin Scrumin jälkitarkastelua. Lisäksi yrityksessä on otettu käyttöön viikoittaiset kehityspalaverit, joissa työntekijöille annetaan tilaisuus kertoa asioista, jotka vaatisivat parantamista. Tämä on osoittautunut hyväksi tavaksi kehittää yrityksen toimintaa, ja yrityksen johto on pyrkinyt reagoimaan kehitysideoihin. (15.4.2011, haastattelu.)

Ohjelmistokehittäjien samanaikainen työskentely useassa eri projektissa tuo Päivän Scrumin käytäntöjä noudattaviin palavereihin omat haasteensa. Kun jokin projekteista on sellaisessa vaiheessa, että tiimin jäsenet odottavat toisen tiimiläisen toteuttaman ohjelmiston osan valmistumista ennen kuin voivat jatkaa omaa työtään, ovat aamupalaverit hyvä tilaisuus päivittää projektin tilannetta toisille tiimin jäsenille. Yrityksen projektikäytännöistä johtuen samassa palaverissa käsitellään kuitenkin väistämättä usean eri projektin tilannetta, jolloin myös ne työntekijät, jotka eivät työskentele kyseisessä projektissa, joutuvat käyttämään aikaansa omaan työhönsä liittymättömien asioiden kuuntelemiseen. (Ohjelmistosuunnittelija 18.4.2011, haastattelu.) Tällöin aamupalavereista voi muodostua helposti tilaisuuksia, joissa käydään läpi ohjelmistokehittäjän ja projektipäällikön kahdenvälisiä asioita. Toisaalta työntekijät eivät välttämättä keskity kuuntelemaan sellaista tietoa, joka ei tunnu heistä oleelliselta heidän oman työnsä kannalta (Ohjelmistosuunnittelija 18.4.2011, haastattelu). Tässä on vaarana se, että jotain tärkeää tietoa jääkin kuulematta.

Vaikka joitakin Scrumin käytäntöjä on alettu jo hyödyntää, ei yrityksessä ole projektipäällikön mukaan vielä mietitty, kuka ottaa hoitaakseen Scrum-mestarin tehtävät. Tällä hetkellä hän kokee vastaavansa niistä tehtävistä, joita Scrum-mestarilla on. Hän näkee ongelmana kuitenkin sen,

että hän ei muista vastuualueistaan johtuen voi olla tiiviisti mukana ohjelmistokehitystiimissä. Hänen näkemyksensä mukaan ideaalitilanne olisi, että yksi projektitiimin jäsenistä toimisi myös Scrum-mestarina. Tälle työntekijälle varattaisiin aikaa sekä ohjelmistokehitykseen että Scrum-mestarin tehtävien hoitamiseen. (15.4.2011, haastattelu.)

Projektipäällikkö totesi, että yritys ei halua päästää asiakasta mukaan Scrum-tiimiin. Toimeksiantajayrityksessä tiedostetaan Scrumin tuoteomistaja-rooli, mutta sen ei nähdä sopivan sellaisenaan yrityksen projektimalliin. Tällä hetkellä myyjä, joka on myynyt kehitettävän ohjelmiston asiakkaalle, toimii asiakkaan ja tiimin välissä ja hoitaa asiakkaan roolia. Tilaaja-asiakas ei osallistu kunkin sprintin aikana toteutettavien ominaisuuksien valintaan. Tämä koetaan toimeksiantajayrityksessä ongelmana, koska valitut ominaisuudet eivät välttämästä vastaa asiakkaan toivetta siitä, mitä tämä haluaisi kehitettävän ensimmäisenä. (15.4.2011, haastattelu.)

Yritys on sitoutunut ja varautunut Scrumin mahdollisen käyttöönoton aiheuttamiin kustannuksiin. Projektinhallinnan kehittämiseen liittyvät kulut hyväksytään, koska odotettavissa on hyvä lopputulos tehostuneen toiminnan ja laadukkaampien tuotteiden muodossa. Ulkopuolista konsulttia ei haluta hyödyntää, vaan käyttöönotosta aiotaan selviytyä yrityksen omilla resursseilla. Yrityksessä hyväksytään myös vaihtoehto, jossa Scrumin käyttöönottoa ei toteutetakaan, jos siitä ei koeta saatavan riittävää hyötyä. Prosessin aikana saatu tietämys ja oppi koetaan tappion sijasta arvokkaana. (Projektipäällikkö 15.4.2011, haastattelu.)

Projektinhallinnan kehittämisestä toivotaan toimeksiantajayrityksessä apua niin projektin kuin lopputuloksenkin laadun parantamiseen. Toiveena on, että projektin tehokkaamman hallitsemisen sekä hyvin jaettujen roolien ja tehtävien avulla voidaan pysyä paremmin aikatauluissa. Tehokkuutta toivotaan löytyvän oikealla tavalla toimimisesta ja oikeanlaisten menetelmien käyttämisestä. (Projektipäällikkö 15.4.2011, haastattelu.)

Yrityksessä halutaan kehittää projektin loppuvaiheen dokumentointia. Projektipäällikkö on huomannut, että projektin alussa dokumentoinnista huolehditaan, mutta loppuraportointi jää usein tekemättä. Hänen mielestään sen avulla saataisiin karsittua mahdollisesti myöhemmin havaittavia, keskeneräisiksi jääneitä osa-alueita. Loppuyhteenvedon avulla voidaan hänen näkemyksensä mukaan parantaa projektin laatua, tarkastaa lopputulos sekä kehittää tehtävien määrittelyä ja aika-arviointia. Tällöin voidaan oppia uusia asioita seuraavaa projektia varten. (15.4.2011, haastattelu.)

5.2 Toimeksiantajayrityksen ohjelmistokehitysmenetelmän kehittäminen

Kuten Leffingwell ja Smits tuovat esille, Scrumin käyttöönottoa ei voida suunnitella ja toteuttaa tehtävälistojen, proseduurien ja lomakkeiden avulla. Jokainen organisaatio on erilainen, eikä huolellisella etukäteissuunnittelulla voida välttyä käyttöönotossa eteen tulevista hankaluuksista ja vaikeuksista. (2005, 9.) Näin ollen toimeksiantajan ohjelmistokehitysmenetelmän kehittämisestä ei tehdä kokonaisvaltaista suunnitelmaa tämän opinnäytetyön puitteissa. Toimeksiantajan edustajille tehtyjen haastatteluiden tulosten ja opinnäytetyön tekijän tekemien havaintojen avulla nostetaan kuitenkin esille yrityksen ohjelmistokehityksen kehittämiskohteita ja ehdotetaan niihin Scrumin tai muiden menetelmien käytäntöjen mukaisia parannuksia.

Schwaberin mukaan Scrumin käyttöönotto vaatii yrityksen johdolta aikaa ja vaivaa. Mikäli toimeksiantajayritys haluaa siirtyä Scrumin käyttöön, sen johdon tulee ymmärtää, että tämä prosessi ei käy hetkessä muiden töiden ohessa, vaan vaatii järjestelmällistä ja huolellista lähestymistapaa. (2007, 5.) Myös työntekijöillä tulee olla mahdollisuus irrottautua joiksikin päiviksi muista töistä ja paneutua kunnolla Scrumiin yhdessä yrityksen johdon kanssa. Käyttöönottoon tuleekin lähteä vain siinä tapauksessa, että yrityksellä on siihen tarvittavat resurssit ja yrityksen johto on vakuuttunut vaivannäön kannattamisesta (Schwaber 2007, 5). Toimeksiantajayrityksen johdon kannattaakin vielä pohtia, mitä konkreettista hyötyä Scrumin käyttö tuo yritykselle.

Leffingwell ja Smits korostavat, että koko organisaation tulee ottaa vastuu sekä Scrumin periaatteiden mukaisesti toimimisesta että eteen tulevien kehitystyötä estävien tekijöiden poistamisesta. Molemmat ovat haastavia asioita ja vaativat kovaa työtä varsinaisen ohjelmiston kehitystyön lisäksi. (2005, 8.) Haastatteluiden ja toimeksiantajayrityksen toimintaan tutustumisen perusteella voidaan todeta, että yrityksellä ei tällä hetkellä ole aika- ja henkilöresursseja viitekehyksessä esitellyn laajamittaisen Scrumin käyttöönottoprosessin käynnistämiseen. Yrityksen johdolla ei ole aikaa käyttöönoton vaiheiden läpiviemiseen eikä yrityksessä ole tällä hetkellä henkilöä, jolla olisi aikaa vastata kaikista organisaatiotasoisen Scrum-mestarin tehtävistä. Scrumista ja muista ohjelmistokehitysmenetelmistä voidaan kuitenkin ottaa käyttöön joitakin yksittäisiä projektitoimintaa edistäviä käytäntöjä. Tällä tavoin voidaan saavuttaa uudenlaista tehokkuutta ja parantaa ohjelmistotuotteiden laatua. Toimeksiantajayrityksessä voidaan harkita Scrumin täysipainoisempaa hyödyntämistä tulevaisuudessa, kun tilanne on siihen otollisempi ja yritys työllistää mahdollisesti useampia ohjelmistokehitykseen osallistuvia ammattilaisia.

Yrityksen ohjelmistokehitysprosessia voidaan kehittää, kun sekä johto että työntekijät ovat sitoutuneita muutokseen. Haastatteluiden ja opinnäytetyön tekijän omakohtaisten kokemusten perusteella voidaan todeta, että toimeksiantajayrityksen johto on sitoutunut projektinhallinnan kehittämiseen enemmän kuin työntekijät. Tämä voi johtua siitä, että muutoksia tuodaan yritykseen johdon aloitteesta, ylhäältä alaspäin, eikä käyttöönotettavista menetelmistä ja ohjeistuksista keskustella työntekijöiden kanssa riittävässä määrin. Työntekijät tuovat kyllä esille omia mielipiteitään toteutettuihin projekteihin ja niiden lopputuloksiin liittyen, mutta heille ei välttämättä ole muodostunut kokonaisnäkemystä projektinhallinnan kehittämisen tuomista eduista. Tähän tulee toimeksiantajayrityksessäkin kiinnittää huomiota.

Päätetäänpä yrityksessä hyödyntää Scrumia kuinka pienimuotoisesti tai laajasti hyvänsä, tulee yrityksessä päättää siitä, kuka toimii Scrum-mestarina uusien käytäntöjen käyttöönoton aikana ja sen jälkeen. Kuten työn viitekehyksessä kerrotuista asioista on havaittavissa, Scrum-mestarin osuus on ratkaisevan tärkeä. Hänen tulee olla tehtäviensä tasalla, sillä uusien käytäntöjen käyttöönotto onnistuu sitä paremmin ja tehokkaammin, mitä tiiviimmin Scrum-mestari voi olla mukana opastamassa ja valmentamassa kehitystiimin toimintaa sekä auttamassa ongelmien ja konfliktien ratkaisemisessa ja poistamassa käyttöönoton onnistumisen eteen tulevia seikkoja. Tähän rooliin ei pidä suhtautua kevyesti, ja rooliin valitulla henkilöllä tulee olla riittävästi aikaa ja motivaatiota tehtäviensä hoitamiseen. Ketään ei sovi vain määrätä tai velvoittaa toimimaan Scrum-mestarina, vaan hänellä tulee olla aito halu toimia tehtävässä.

Toimeksiantajayrityksessä tulee puuttua projektipäällikön (15.4.2011) haastattelussa esiin tulleeseen Scrumin käytäntöjen vastaiseen tilanteeseen: tällä hetkellä sama henkilö – projektipäällikkönä toimiva yrittäjä – vastaa sekä Scrum-mestarin että tuoteomistajan tehtävistä. Sekä Deemerin (2010, 7) että Schwaberin ja Sutherlandin (2010, 7) mukaan Scrum-mestari ja tuoteomistaja eivät saa olla sama henkilö, joten toimeksiantajayrityksen käytäntö on selkeästi ristiriidassa Scrumin periaatteiden kanssa. Tämä ristiriita voidaan ratkaista siirtämällä Scrum-mestarin tehtävät jonkun kehitystiimiläisen vastuulle. Näin nykyinen projektipäällikkö voi hoitaa yhteydenpidon varsinaisiin asiakkaisiin ja toimia projekteissa eräänlaisena tuoteomistajana. Tämä ei vastaa täysin Scrumin periaatteita, mutta selkiyttää Scrum-tiimin roolijakoa. Tällä hetkellä haasteena on kuitenkin se, että kaikkia pienen kehitystiimin jäseniä tarvitaan ohjelmistosuunnittelun ja ohjelmoinnin parissa. Tilanne voi helpottua, mikäli yritys päätyy palkkaamaan myöhemmin lisää työvoimaa.

Mikäli Scrum-mestarin tehtävät päätetään siirtää jonkun kehitystiimin jäsenen vastuulle, kannattaa miettiä huolella, kuka tiimin jäsenistä ottaa roolin hoitaakseen. Scrum-mestarilla tulee olla tehtäviensä hoitamiseen riittävästi aikaa etenkin, kun yrityksessä tai tiimissä ollaan ottamassa käyttöön jotain Scrumin käytäntöä, sekä jatkossa erityisesti sprintin alussa ja lopussa (Adkins 2010, 78). Jos yrityksen kokenein ohjelmistosuunnittelija toimii Scrum-mestarina, hänen aikaisempiin vastuualueisiinsa antama työpanos pienenee. Tällöin esimerkiksi tiimin tuottaman koodin määrä vähenee voimakkaammin kuin tilanteessa, jossa Scrum-mestarin tehtäviä alkaa hoitaa joku ohjelmoinnin saralla kokemattomampi tiimin jäsen. Olipa Scrum-mestarin roolia hoitava ohjelmistosuunnittelija kuka tahansa, käytännön ohjelmistokehitystyöhön ei jää enää niin paljon aikaa. Hänen ohjelmointiin ja ohjelmistosuunnitteluun liittyvää ammattitaitoaan voidaan kuitenkin hyödyntää edelleen esimerkiksi puolipäiväisesti ja lisäksi hän voi opastaa muita tiimiläisiä.

Jos Scrum-mestariksi ehdotettu ohjelmistokehittäjä ei ole riittävän motivoitunut uusiin tehtäviinsä, tulee rooliin harkita jotain toista tiiminjäsentä. Jos yrityksen työntekijöillä ei ole kiinnostusta tai tarvittavaa tietotaitoa Scrum-mestarin tehtävien hoitamiseen, voidaan harkita uuden, projektinhallintaan perehtyneen työntekijän palkkaamista. Hänellä tulisi olla osaamista myös ohjelmistosuunnittelusta, sillä toimeksiantajayrityksessä ei ole tällä hetkellä tarvetta pelkästään Scrum-mestarina kokoaikaisesti toimivalle työntekijälle. Uusi työntekijä voi tuoda yritykseen uusia näkökulmia. Hänellä tulee olla tarvittava rohkeus asioiden kehittämiseen ja Scrumin mukaisten käytäntöjen läpiviemiseen.

Scrum-mestarin tehtävien siirtyminen nykyiseltä yrittäjä-projektipäälliköltä eri työntekijälle muuttaa yrityksen nykyisiä toimintatapoja. Projektipäällikön tulee siirtyä perinteisestä johtajan roolista taustavaikuttajan rooliin, hoitaa tuoteomistajan tehtäviä ja antaa valitulle Scrum-mestarille ja kehitystiimille vastuu projektien onnistumisesta. Toki hänen tulee yrityksen johtajana olla tietoinen Scrum-tiimin toiminnan tuloksista ja tuottavuudesta, mutta hänen tulee päättää siirtyä "siasta" "kanaksi". Tämä vaatii uskalluksen lisäksi luottamusta Scrum-tiimin jäsenten ammattitaitoon.

Virtasen (17.6.2010, luento) mukaan asiakkaan edustajan tulee olla tuoteomistajana asiakasprojekteissa. Projektipäällikön haastattelussa kävi ilmi, ettei yritys ole valmis ottamaan asiakasta tuoteomistajan rooliin, vaan yrityksen johto haluaa pitää asiakkaan erillään ohjelmistokehitysprojektin käytännön työstä. Scrumin mukaisesti toimittaessa tiimin jäsenet keskustelevat asiakkaan kanssa selvittääkseen tuotteelta toivotut ominaisuudet (Schwaber & Beedle 2001, 21). Yrityksessä kannattaa vielä harkita asiakkaan edustajan tuomista projektiin tuoteomistajan

roolissa, sillä tällä voidaan vaikuttaa kehitystiimin käsitykseen asiakkaan toiveista ja halutusta lopputuloksesta.

Kun asiakkaan edustaja ei toimi tuoteomistajana, tieto asiakkaan odotuksista, toiveista ja vaatimuksista kulkee välikäden kautta. Tällöin on tärkeää pitää huoli siitä, että kehitystiimi saa muodostettua hyvän käsityksen siitä, mitä asiakas oikeasti haluaa. Ainakin jotkut kehitystiimin jäsenistä voisivat olla mukana osassa asiakastapaamisista. Tällöin myös kehitystiimi saa konkreettista tuntumaa asiakkaan näkökulmasta, ja osaa toimia asiakaslähtöisemmin esimerkiksi projektin aikaisen testauksen yhteydessä. Kehitystiimin ja asiakkaan edustajan yhteistyöllä voidaan myös vakuuttaa asiakas siitä, että projektiin osallistuvat ohjelmistokehittäjät ovat oman alansa asiantuntijoita ja siten päteviä suunnitelman toteuttamiseen (Schwaber 2004, 68). Mikäli kehitystiimin jäsenillä ei ole kokemusta asiakastapaamisista, heitä voidaan opastaa yrityksen johdon toimesta. Tällöin he saavat käsityksen siitä, miten asiakkaan kanssa toimitaan ja mitä asioita voidaan nostaa esille. Yhteisen kielen käyttäminen on myös tärkeää ja Schwaberin (2004, 65–66) mukaan kehitystiimin jäseniä tulee opastaa käyttämään asiakkaan edustajalle tuttua kieltä. Asiakkaan kanssa ei tule käyttää liian teknistä sanastoa, vaan asiakasta tulee lähestyä hänen näkökulmastaan ja käsitemaailmastaan.

Ohjelmistosuunnittelijan (18.4.2011, haastattelu) mukaan toimeksiantajayrityksen asiakkailla ei välttämättä ole aluksi hyvää kuvaa haluamastaan lopputuloksesta. Tällöin kehitystiimillä voi olla vaikeuksia hahmottaa, millaisista ratkaisuista olisi eniten hyötyä asiakkaalle. Scrumin käytäntöjä noudatettaessa jo ensimmäisen sprintin aikana toteutetaan jokin ohjelmiston osa, joka esitellään asiakkaalle sprintin katselmoinnissa. Toisinaan voi kuitenkin olla hyödyksi selvittää vaatimuksia tarkemmin ennen ensimmäiseen sprinttiin etenemistä.

Tällaisessa tapauksessa voidaan harkita luvussa 2.1.3 esitellyn protoilumallin mukaista prototyypin tekemistä jostain ohjelmiston avaintoiminnosta. Prototyyppi voi olla hyvin yksinkertaistettu ja keskeneräisenkin oloinen ohjelmiston osa, joten se voidaan toteuttaa nopeasti. Prototyypin avulla asiakkaalta saadaan käytännönläheisempää palautetta, jonka pohjalta suunnitelmia voidaan tarkentaa. Prototyyppi voidaan tehdä ohjelmoidun proton lisäksi esimerkiksi paperiversiona, jossa asiakkaalle ja loppukäyttäjille esitellään ohjelmiston tietyn osion toiminnallisuutta ja käyttöliittymähahmotelmaa. Näin asiakkaan edustaja saa myös konkreettisemman käsityksen toteutettavasta ohjelmistosta, hän kykenee ideoimaan toteutettavaa ohjelmistoa ja voi antaa

suoraa palautetta siitä, mikä toimii ja mihin hän haluaisi muutoksia. Tällä tavoin asiakas saadaan myös sitoutumaan paremmin projektiin.

Toimeksiantajayrityksen projektitiimien henkilömäärä vaihtelee kahden ja neljän välillä (Projektipäällikkö 15.4.2011, haastattelu). Näin pienet tiimit aiheuttavat haasteita Scrumin tai minkä tahansa muunkin ohjelmistokehitysmenetelmän mukaisesti toimittaessa. Schwaberin ja Sutherlandin mukaan ihanteelliseen kehitystiimiin kuuluu viidestä yhdeksään jäsentä. Heidän mukaansa pienemmästä tiimistä ei välttämättä löydy tarvittavaa osaamista jonkin ominaisuuden toteuttamiseksi, ja lisäksi vuorovaikutus ei ole yhtä tehokasta kuin ihannekokoisessa tiimissä. (2010, 8.) Toimeksiantajayrityksen tilanteessa voidaan tehdä johtopäätös, että projekteissa, joiden toteuttamiseen osallistuu vain yksi ohjelmistosuunnittelija, on mahdotonta toteuttaa mitään tiimeihin pohjautuvaa ohjelmistokehitysmenetelmää. Tällaisten projektien kannalta tehokkainta voi olla se, että asiakkaan vaatimusten huolellisen selvittämisen jälkeen ohjelmistosuunnittelija tekee itsenäisesti karkeat toteutussuunnitelmat ja kehittää niitä lisää projektin edetessä. Onneksi yrityksessä on kuitenkin sellainen henki, että työntekijät neuvovat auliisti toisiaan. Tämä auttaa projekteissa, joissa työskentelee projektipäällikön lisäksi vain yksi henkilö.

Toimeksiantajayrityksessä tiimihenki ja yhteistyö ovat hyvällä tasolla. Ohjelmistosuunnittelijan (18.4.2011) ja projektipäällikön (15.4.2011) haastatteluissa tuli kuitenkin ilmi joitakin kommunikaatiosta johtuvia ongelmia. Tiimin jäsenet tekevät esimerkiksi yksittäisten ominaisuuksien tai osa-alueiden toteuttamisessa itsenäisiä ratkaisuja. Nämä ratkaisut voivat vaikuttaa toistenkin tiimiläisten tekemiin osa-alueisiin, joten toteutustavoista yhteisesti sopiminen on erittäin tärkeää. Kehitystiimin tuleekin varoa, että ohjelmistokehityksessä ei sorruta luvussa 2.1.1 käsitellyn koodaa ja korjaa -mallin mukaiseen toimintaan. Projektien aikana tulee toteuttaa vain ohjelmiston arvoa parantavia ominaisuuksia ja toimintoja. Ohjelmoijien tulee varoa innostumasta omien visioidensa toteuttamisesta, jos ne eivät ole linjassa muun toteutuksen kanssa. Tämä voidaan välttää sillä, että ohjelmoijat pitävät mielessä ohjelmistolle asetetut vaatimukset ja yhteiset suunnitelmat. Suunnitelmia noudattamalla voidaan säästyä turhilta työtunneilta ja korjauksilta.

Schwaber ja Beedle (2001, 152) kertovat, että ihannetilanteessa tiimin jäsenet muodostavat toisiaan kunnioittavan ja itseohjautuvan kehitystiimin. Tiimi ei kuitenkaan välttämättä osaa sisäistää tätä asiaa itsenäisesti, vaan tarvitsee Scrum-mestarin ohjausta (Tieturi Oy 2010, 64.) Schwaberin (2004, 101) mukaan kokeneillakin IT-alan ammattilaisilla voi olla ongelmia ensimmäisten sprinttien aikana. Onkin ensiarvoisen tärkeää, että käyttöönotosta vastaava Scrum-

mestari kannustaa tiimiä Scrumin arvojen soveltamiseen, esimerkiksi mielipiteiden rohkeaan esilletuontiin. Toimeksiantajayrityksessä tulee huomioida myös Schwaberin (2004, 30) korostama asia, jonka mukaan itseohjautuva kehitystiimi ei tarkoita sitä, että Scrum-mestarin tiivistä panosta ei tarvita projektien aikana.

Scrumin mukaisia toimintatapoja käyttöön otettaessa työntekijöitä tulee perehdyttää riittävästi uusiin käytäntöihin. Kehitystiimin jäsenten tulee ymmärtää, mitä heiltä odotetaan, mihin he ovat sitoutumassa ja mitä hyötyä tästä on yrityksen ja tiimin toiminnalle. Tällä voidaan lisätä tiimin jäsenten motivaatiota muutosten läpikäymiseen ja sen yhteydessä tulevien ongelmien ratkaisuun. Motivaatiota parantaa myös se, jos muutokset ovat lähtöisin kehitystiimiläisiltä itseltään. Projektipäällikkö (15.4.2011, haastattelu) kertoi, että yrityksessä pyritään reagoimaan työntekijöiltä tulleisiin kehitysideoihin, ja tätä samaa voidaan soveltaa myös Scrumin käyttöönoton yhteydessä.

Toimeksiantajayrityksessä on otettu käyttöön joitakin Scrumin mukaisia käytäntöjä, ja niiden toimivuutta on jo kehitetty edelleen. Tässä on havaittavissa selvää itseoppimista sekä Scrum-mestarin roolia tällä hetkellä hoitavan projektipäällikön että kehitystiimin osalta. Vuoden 2010 alussa Päivän Scrumeissa oli huomattavissa, että osallistujat eivät olleet vielä käsittäneet palavereiden merkitystä. Muut tiimiläiset eivät välttämättä kuunnelleet kulloinkin äänessä olevaa tiimin jäsentä, vaan jatkoivat työntekoa kunnes tuli heidän vuoronsa vastata Päivän Scrumin kolmeen kysymykseen. Näin toimittaessa päivittäiset palaverit tukivat enemmän projektipäällikön ja yksittäisen työntekijän välistä tiedonvälitystä, eivätkä aidosti parantaneet tiimin välistä kommunikaatiota. Yrityksessä on kuitenkin selvästi huomattu näiden palavereiden tärkeys, ja tällä hetkellä Päivän Scrumit palvelevat paremmin tarkoitustaan. Haastateltu projektipäällikkö (15.4.2011, haastattelu) kertoi, että yrityksessä huolehditaan palavereiden säännöllisestä pitämisestä, ja ohjelmistosuunnittelija (18.4.2011, haastattelu) toi esille Päivän Scrumien merkityksen tiimin sisäisen tiedonkulun parantumisessa. Tämä esimerkki kertoo siitä, että yrityksen työntekijätkin ovat alkaneet nähdä Scrumin käytön tuomia hyötyjä, eikä sen tuomia käytäntöjä pidetä vain välttämättömänä pahana saati työntekoa haittaavina tekijöinä.

Projektipäällikön (15.4.2011) haastattelussa kävi ilmi, että toimeksiantajayrityksessä pidetään jokaisen projektin alussa Scrumin käytäntöjä noudatteleva suunnittelupalaveri, jossa toteutettava ohjelmisto jaetaan työtehtäviin. Tässä voidaan hyödyntää käyttäjätarinoita, jolloin saadaan lisää selkeyttä siihen, mitä käyttäjän tulee voida tehdä ohjelmistossa. Näin voidaan välttyä tilanteelta, jossa jonkin ohjelmiston osan ajatellaan olevan valmis, mutta testauksen aikana huomataankin,

että suunnitelmissa ei ole osattu ottaa huomioon jotain toimintoa. Käyttäjätarinat voivat selkeyttää tekemistä alusta alkaen. Ne auttavat ominaisuuksien hahmottamisessa, työaikamäärien arvioimisessa ja eri toimintojen käyttäjälähtöisessä testaamisessa. Niiden avulla saadaan myös muodostettua käsitys siitä, missä järjestyksessä ohjelmiston ominaisuudet ja osat kannattaa tehdä. Erityisesti ei-toiminnalliset vaatimukset ovat sellaisia, jotka pitää ottaa huomioon ennen muiden vaatimusten toteuttamista. Tällöin ne osa-alueet, jotka vaikuttavat muihin ohjelmiston osioihin, mietitään valmiiksi heti projektin alussa, ja toteutetaan ennen kuin muita osioita aletaan työstää. Näin voidaan välttyä myöhemmin tehtäviltä, aikaa vieviltä korjauksilta.

Projektipäällikön (15.4.2011) haastattelussa nousi esille tarve projektien aikataulutuksen kehittämiseen ja tehtävien edistymisen seurannan parantamiseen. Yritys käyttää projektinhallinnan ohjelmistoa, johon kirjataan työtehtävien etenemiseen liittyviä tietoja. Sen kautta ei kuitenkaan saada tietoa siitä, kuinka paljon sprintin aikana tehtävien ominaisuuksien toteuttamiseen on käytettävissä aikaa. Tämä aiheuttaa sen, että projektin todellisesta tilanteesta ei ole tietoa. Tätä voidaan kehittää ottamalla yrityksessä käyttöön taulukkomuotoinen tuotteen työlista ja sprintin tehtävälista sekä niihin liittyvät edistymiskäyrät. Tämä voidaan toteuttaa esimerkiksi taulukkolaskentaohjelmalla, johon tehdään valmiit työ- ja tehtävälistapohjat, joita päivittämällä ylläpidetään myös graafisia edistymiskäyriä. Yrityksessä voidaan hyödyntää myös Internetistä löytyviä mallipohjia tai muita tämän asian toteuttamiseen tehtyjä sovelluksia.

Projektipäällikkö (15.4.2011, haastattelu) kertoi, että yrityksestä ei löydy projektin johtamiseen perehtynyttä henkilöä eikä yrityksessä näin ollen ole hyödynnetty mitään tiettyä ohjelmistokehitysmenetelmää. Mikäli yrityksen projekteja olisi totuttu hallitsemaan jonkin tietyn menetelmän mukaisesti, saattaisi tästä olla haittaa uuden menetelmän mukaisia käytäntöjä käyttöönotettaessa. Leffingwellin ja Smitsin (2005, 5) mukaan uuden ohjelmistokehitysmenetelmän käyttöönottaminen vaatii aina avointa suhtautumista ja kyseisen menetelmän perusperiaatteiden ymmärtämistä. Näin olleen voidaankin katsoa jopa eduksi, että yrityksen projektinhallinnassa ei ole käytetty mitään tiettyä menetelmää, jolloin ohjelmistoprojektien kehittämiseen voidaan lähteä avoimin mielin.

Toimeksiantajayrityksen yrittäjät (joista toinen toimii myös projektipäällikkönä) ja yksi ohjelmistosuunnittelijoista on osallistunut kaksipäiväiseen ScrumMaster-koulutukseen (projektipäällikkö 15.4.2011, haastattelu). Tämä on hyvä alku, mutta lisäkoulutus on tarpeen, mikäli Scrum otetaan tulevaisuudessa laajemmin käyttöön. Ohjelmistokehitystiimin kaikilla jäsenillä tulee olla tarvittavat

tiedot käytettävästä menetelmästä, jotta he osaavat toimia Scrumin mukaisesti. Oppia voidaan hakea itsenäisesti esimerkiksi Internetistä tai kirjoista, mutta lisäksi kannattaa selvittää ulkopuolisen kouluttajan järjestämien kurssien mahdollisuudet. Mikäli kaikki työntekijät eivät ota osaa kurssiin, kurssille osallistuneen tulee jakaa tietonsa muille yrityksen sisäisessä koulutustilaisuudessa. Suomessa on saatavilla Scrum-mestarin koulutuksen lisäksi esimerkiksi yleistä ketteriin menetelmiin ja Scrumiin liittyvää koulutusta sekä ketterän menetelmän käyttöönottoon ja tuoteomistajan tehtäviin liittyvää koulutusta. Tietoa koulutuksia löytyy esimerkiksi Scrum Alliancen tai Tieturi Oy:n Internet-palvelun kautta.

Haastatteluissa kerrottiin, että yrityksen ohjelmistosuunnittelutiimin jäseniltä löytyy koulutusta muista ohjelmistokehityksen osa-alueista testausta lukuun ottamatta (projektipäällikkö 15.4.2011, haastattelu; ohjelmistosuunnittelija 18.4.2011, haastattelu). Pienessä tiimissä testauksen haasteellisuutta lisää se, että jokainen ohjelmoija joutuu testaamaan myös omaa koodiaan. Ketterää menetelmää hyödynnettäessä testausta ei tule tehdä vain projektin loppuvaiheessa, vaan jatkuvasti sen aikana. Tällä edesautetaan käsitystä projektin todellisesta tilanteesta ja toteutetaan Scrumin arvoa avoimuudesta. Jo näiden seikkojen vuoksi yrityksen testauskäytäntöjen kehittämiseen tulee kiinnittää huomiota. Scrum ei tarjoa tähän teknisiä ratkaisuja, joten yrityksen kannattaa etsiä apua testauskäytäntöihinsä muista menetelmistä. Yrityksessä voidaan tutustua esimerkiksi Extreme Programming -menetelmän tai ketterien testaustapojen tarjoamiin ratkaisuihin, joiden joukosta voi löytyä toimeksiantajalle sopivia käytäntöjä. Myös automaattisiin testausympäristöihin kannattaa tutustua.

Toimeksiantajayrityksessä työskentelee osa-aikaisia työntekijöitä ja myös opiskelijoille tarjotaan harjoittelupaikkoja (Projektipäällikkö 15.4.2011, haastattelu). Tämä johtaa siihen, että kehitystiimin jäsenet vaihtuvat aika-ajoin. Virtasen (17.6.2010, luento) mukaan kehitystiimin jäsenten toisiinsa tutustumista voidaan helpottaa niin sanotun kick off -päivän avulla. Toimeksiantajayrityksessä voidaan ottaa tavaksi yhteinen ajanviettotilaisuus aina, kun yritykseen tulee uusi ohjelmistokehittäjä. Tilaisuus voi olla esimerkiksi muutaman tunnin mittainen lounas tai saunailta. Tällöin työntekijät voivat tutustua toisiinsa ja tällä voidaan myös edesauttaa Schwaberin ja Beedlen (2001, 147) esittelemien kunnioituksen ja rohkeuden arvojen toteutumista. Tiimin jäsenille muodostuu käsitys toistensa taustasta, koulutuksesta ja työkokemuksesta. Kun uusi työntekijä kokee olevansa tervetullut, hän saa rohkeutta täysivaltaisena tiimin jäsenenä toimimiseen ja uskaltaa myös pyytää apua sitä tarvitessaan.

6 POHDINTA

Opinnäytetyöni tavoitteena oli tutustua ohjelmistokehitysmenetelmistä erityisesti Scrumiin ja sen käyttöönottoon, ja käsitellä miten hyvin Scrum sopii toimeksiantajayritykselle ja miten yrityksen ohjelmistokehityskäytäntöjä voidaan parantaa. Kehittämistehtävän tueksi haastattelin kahta toimeksiantajayrityksen edustajaa. Mielestäni haastattelut onnistuivat hyvin, ja sain hyvän käsityksen yrityksen nykytilanteesta ja kehittämistarpeista. Kehittämistehtävän päätuloksena totesin, että tällä hetkellä laajamittaisen Scrumin käyttöönottoprosessin käynnistäminen ei ole realistista, sillä toimeksiantajayrityksellä ei ole käytössä tarvittavia resursseja eikä Scrum sovi kaikilta osin yrityksen nykytilanteeseen. Toin kehittämistehtävässä kuitenkin esille erilaisia pienimuotoisempia ehdotuksia yrityksen ohjelmistokehitysprosessin parantamiseksi ja kehittämiseksi ketterien menetelmien suuntaan.

Opinnäytetyöprosessini alkupuolella toimeksiantajayrityksessä otettiin käyttöön joitakin Scrumin mukaisia käytäntöjä. Tämä tuli minulle hieman yllätyksenä, mutta ei lopulta vaikuttanut opinnäytetyöni kehittämistehtävään. Huomioin yrityksen edustajien haastattelukysymyksissä heidän sen hetkiset Scrum-käytäntönsä, ja kehittämistehtävässä otin kantaa niiden toimivuuteen.

Opinnäytetyön aiheen rajaus onnistui lähes alkuperäisten suunnitelmien mukaisesti. Toimeksiantaja toivoi ohjelmistokehitysmenetelmäänsä ratkaisua, jossa Scrumiin on yhdistetty muiden menetelmien käytäntöjä. Niinpä käsittelin raportin toisessa luvussa lyhyesti perinteisiä ja ketteriä menetelmiä, jotta sekä minä että toimeksiantaja saimme käsityksen erilaisten menetelmien kirjosta. Kokonaisvaltaisen ohjelmistokehitysmenetelmän laatiminen toimeksiantajayritykselle olisi vaatinut syvempää perehtymistä erilaisiin menetelmävaihtoehtoihin, mikä osoittautui liian laajaksi tehtäväksi ammattikorkeakoulun opinnäytetyön puitteissa. Näin ollen keskityin pohtimaan Scrumin eri käytäntöjen soveltuvuutta yrityksen käyttöön, enkä käsitellyt muiden menetelmien tarjoamia ratkaisuja kuin muutamin osin.

Kehittämistehtävän päätuloksesta johtuen en voinut hyödyntää Scrumin käyttöönotosta kertovaa lukua täysipainoisesti toimeksiantajayrityksen ohjelmistokehitysmenetelmiä pohtiessani. Luvusta on yritykselle kuitenkin hyötyä, jos he päättävät ryhtyä Scrumin käyttöönottoon myöhemmässä vaiheessa. Tällöin yritys voi noudattaa pilottiprojektin suunnittelusta liikkeelle lähtevää, koko organisaatiota koskevaa käyttöönottoprosessia ja saada Scrumista täysimittaisen hyödyn.

Käyttöönottoon liittyvistä haasteista kertova luku valmentaa yritystä siihen, millaisia esteitä yrityksen johdon ja valitun Scrum-mestarin purettaviksi voi tulla.

Yrityksen edustajien haastatteluissa nousi esille testaukseen liittyviä ongelmia ja haasteita. Näihin asioihin ei tämän työn puitteissa voitu kuitenkaan mennä syvällisemmin. Yrityksessä tulisi tutustua erilaisiin testausmenetelmiin ja niiden soveltuvuuteen. Mikäli tämä ei onnistu omilla resursseilla, voidaan yrityksessä tarjota aihetta esimerkiksi ammattikorkeakoulun opiskelijan opinnäytetyöksi.

Opinnäytetyön aihealue ei ollut minulle entuudestaan tuttu, mikä toi työn toteuttamiseen mielenkiintoa, mutta myös haasteita. Työn alkuvaiheessa haasteita aiheutti esimerkiksi eri termien ymmärtäminen, sillä eri lähteissä samasta termistä käytettiin eri ilmaisuja. Lisäksi minun piti tehdä päätökset siitä, mitä suomenkielisiä termejä käytän työssäni, sillä tästäkin oli erilaisia käytäntöjä.

Opinnäytetyön lähdeaineisto pohjautui enimmäkseen englanninkielisiin lähteisiin. Tämä johtui työn aihepiiristä, josta ei ole julkaistu suomeksi paljoakaan artikkeleita tai kirjoja. Vieraskielisten lähdeaineistojen lukeminen vei ymmärrettävästi enemmän aikaa verrattuna suomenkielisten lukemiseen. Verkkosanakirjat olivat ahkerassa käytössä, vaikkakin pyrin siihen, etten takerru liikaa yksittäisiin sanamuotoihin, vaan yritän ymmärtää sen, mitä kirjoittaja halusi tekstissään tuoda esille. Opinnäytetyöraportin lähdeaineistoihin nojaavien lukujen 2, 3, ja 4 kirjoittamista en kuitenkaan kokenut erityisen vaikeaksi, sillä kun olin ymmärtänyt lähdetekstin, pystyin mielestäni kirjoittamaan varsin sujuvaa, niin sanotusti omin sanoin kerrottua tekstiä.

Yhtenä lähteenä hyödynsin Tieturi Oy:n johtavan konsultin Pentti Virtasen (FT tietojenkäsittely, systeemityön asiantuntija) Certified ScrumMaster -kurssilla kertomia asioita, joita kirjasin ylös kurssille osallistuessani. Koska Virtasen kyseisellä kurssilla kertomista tiedoista ei ole muuta tallennetta kuin omat muistiinpanoni, oli ongelmana lähdekritiikki ja luennon pohjalta kirjoittamieni asioiden paikkansapitävyys. Tämän vuoksi lähetin Virtaselle sähköpostin, jonka liitteeksi laitoin ne opinnäytetyöhöni kirjoittamani kohdat, joissa käytän häntä lähteenä. Virtanen vahvisti, että voin viitata hänen kurssiinsa, joten päätin jättää kokonaisuuden kannalta hyödyllisiksi arvioimani lainaukset opinnäytetyöraporttiini.

Havaitsin, että monet Scrumia ja sen käyttöönottoa esittelevät julkaisut käsittelivät aihetta sellaisen organisaation näkökulmasta, jossa Scrumia hyödynnetään koko organisaation laajuisesti

useissa projekteissa. Kyseisten yritysten projektit ja projektitiimit olivat suurempia kuin toimeksiantajayrityksessä. Tämä aiheutti välillä hieman hankaluuksia pohtiessani, miten sovellan tällaista lähdeaineistoa toimeksiantajan tilanteeseen. Työtä tehdessäni havaitsin kyllä, että monet Scrumin käyttöönottovaiheessa suuremmissa organisaatioissa eteen nousevat asiat tulevat tavalla tai toisella esille myös pienemmissä organisaatioissa.

Moen, Dingsøyrin ja Dybån toteuttama Scrumin käyttöönottoon liittyvä tapaustutkimus on mielestäni neljännen luvun avainlähde. Sen löytäminen lähdeaineistoksi oli erittäin tärkeää käyttöönottoprosessin aikana eteen tulevien käytännön haasteiden ymmärtämiseksi. Näin kyseinen luku ei jäänyt pelkästään teoreettisen tiedon varaan, vaan pohjautui myös käytännönläheiseen tutkimukseen.

Ajanhallinta tuotti minulle välillä hankaluuksia, esimerkiksi kuinka kauan jonkin luvun tai kappaleen kirjoittamiseen tuli varata aikaa. Kirjoittaminen ei tuottanut minulle vaikeuksia, ja keskittyessäni sain tehtyä töitä tehokkaasti. Syvennyin kuitenkin ajoittain liian pitkäksi aikaa johonkin aihepiirin osa-alueeseen, sillä halusin saada siitä perusteellisen käsityksen. Tällöin minun piti keskittyä pohtimaan sitä, olenko jo käsitellyt osa-aluetta riittävästi ja monipuolisesti tärkeimmät näkökannat esille tuoden.

Ajanhallinnan ongelmista huolimatta opinnäytetyö valmistui lähes suunnitellussa aikataulussa. Aloittaessani työtä keväällä 2010 sekä minun että toimeksiantajan tiedossa oli, että olen seuraavan syyslukukauden ajan opiskelijavaihdossa. Halusin vaihdon aikana keskittyä vieraassa maassa opiskeluun ja elämiseen, joten opinnäytetyön tekemiseen tuli silloin tauko. Suomeen palattuani kesti muutaman kuukauden, ennen kuin jatkoin työn parissa, mutta viimeiset kuukaudet työskentelin tehokkaasti ja keskittyneesti. Opinnäytetyö valmistui lopulta vajaan kuukauden alkuperäistä aikataulua myöhemmin eli huhtikuun lopun sijasta toukokuun puolivälissä.

Aikataulun paikkansapitävyyden ja työhön käyttämäni ajan lisäksi olen tyytyväinen kirjoittamaani viitekehykseen ja haastatteluiden onnistumiseen sekä toimeksiantajayrityksen toiminnan kehittämiseksi tekemiini neuvoihin ja ehdotuksiin. Toivon ja uskon työstäni olevan hyötyä toimeksiantajayritykselle: raportin lukemisella yrityksen edustajat voivat perehtyä paremmin Scrumiin ja ohjelmistokehitysmenetelmänsä kehittämisen myötä eteen tuleviin haasteisiin. Yrityksen johto saa uusia näkökulmia ja voi tehostaa yrityksen projektinhallintaa.

Henkilökohtainen tavoitteeni oli saada hyvä käsitys Scrumista ja sen käyttöönoton organisaatiolle asettamista haasteista. Mielestäni saavutin tämän tavoitteen. Lisäksi toimeksiantajan edustajille tekemäni haastattelut antoivat minulle lisäkokemusta käytännön projektityöstä ja sen haasteista. Tämä syventää esimiestyön ja projektinhallinnan opintojaksoilla oppimiani asioita. Ketterät menetelmät ovat IT-alan yrityksissä yleisesti käytetty ohjelmistokehitysmenetelmä, joten koen, että opinnäytetyön toteuttamisen aikana oppimistani asioista on hyötyä tulevalla työurallani.

LÄHTEET

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J. 2002. Agile software development methods: Review and analysis. Hakupäivä 28.2.2011 http://www.vtt.fi/inf/pdf/publications/2002/P478.pdf

Adkins, L. 2010. Coaching Agile Teams: A Companion for ScrumMasters, Agile Coaches, and Project Managers in Transition. Boston: Addison-Wesley.

Beck, K. 1999. Extreme programming explained: embrace change. Hakupäivä 26.3.2011, http://www.mip.sdu.dk/~brianj/Extreme Programming Explained - Kent Beck; Addison-Wesley, 1999.pdf

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D. 2001a. Manifesto for Agile Software Development. Hakupäivä 11.3.2011 http://www.agilemanifesto.org/

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D. 2001b. Principles behind the Agile Manifesto. Hakupäivä 11.3.2011 http://www.agilemanifesto.org/principles.html

Boehm, B. 1988. A Spiral Model of Software Development and Enhancement. Sisäinen lähde. Hakupäivä 3.3.2011 http://csdl.computer.org/dl/mags/co/1988/06/r5061.pdf

Cockburn, A. 2001. Crystal light methods. Hakupäivä 14.3.2011 http://alistair.cockburn.us/Crystal+light+methods

Cockburn, A. 2007. Agile Software Development: The Cooperative Game. Boston: Addison-Wesley.

Deemer, P., Benefield, G., Larman, C. & Vodde, B. 2010. Scrum Primer. Hakupäivä 13.8.2010, http://scrumtraininginstitute.com/home/stream_download/scrumprimer.

Forsberg, K., Mooz, H. & Cotterman, H. 2000. Projektinhallinta – Malli kaupalliseen ja tekniseen menestymiseen. Jyväskylä: Edita.

Haikala, I. & Märijärvi, J. 2006. Ohjelmistotuotanto. Helsinki: Talentum.

Kuikka, S. 2011. Automaation ohjelmistokomponentit ja sovelluspalvelut. Hakupäivä 30.11.2011 https://noppa.tkk.fi/noppa/kurssi/as-116.3172/luennot/AS-116_3172_johdanto.pdf

Leffingwell, D. & Smits, H. 2005. A CIO'sPlaybook for Adopting the Scrum Method of Achieving Software Agility. Hakupäivä 16.8.2010 http://www.leffingwell.org/Document_Store/CIO_Playbook_For_Adopting_Scrum_080805.pdf.

Lindström, J. 2006. Scrum: Scrum-sanastoa. Hakupäivä 30.3.2011 http://www.reaktor.fi/web/fi/teknologia-ja-tutkimus/scrum

Lämsä, A-M & Hautala, T. 2008. Organisaatiokäyttäytymisen perusteet. Helsinki: Edita.

Moe, N. B., Dingsøyr, T. & Dybå, T. 2009. A teamwork model for understanding an agile team: A case study of a Scrum project. Sisäinen lähde. Hakupäivä 2.4.2011 http://www.sciencedirect.com.ezp.oamk.fi:2048/science?_ob=MImg&_imagekey=B6V0B-4XRJX55-1-3&_cdi=5642&_user=965304&_pii=S0950584909002043&_orig=search&_coverDate=11%2F20%2F2009&_sk=999999999&view=c&wchp=dGLzVlz-zSkWA&md5=5c0353d90ce0d1027eadf9ebc619ba75&ie=/sdarticle.pdf.

Ohjelmistosuunnittelija, toimeksiantajayritys. 2011. Haastattelu 18.4.2011. Tekijän hallussa.

Parvikko, O. 2001. Työyhteisön ristiriidat ja konfliktit. Hakupäivä 29.3.2011 http://osha.europa.eu/fop/finland/fi/good_practice/stressi/konflikti.stm

Projektipäällikkö, toimeksiantajayritys. 2011. Haastattelu 15.4.2011. Tekijän hallussa.

Rational Software Corporation. 2001. Rational Unified Process. Hakupäivä 30.3.2011 http://www.cin.ufpe.br/~if682/RUP/

Schach, S. 2007. Object-Oriented & Classical Software Engineering. New York: McGraw-Hill.

Schwaber, K. 2004. Agile project management with scrum. Redmond (WA); Microsoft Press.

Schwaber, K. 2007. The enterprise and Scrum. Redmond (WA): Microsoft Press.

Schwaber, K. & Beedle, M. 2001. Agile software development with scrum. Upper Saddle River: Prentice Hall.

Schwaber, K. & Sutherland, J. 2010. Scrum. Hakupäivä 13.6.2010 http://www.scrum.org/storage/scrumguides/Scrum%20Guide.pdf.

Sininen Meteoriitti. 2011. Refaktorointi. Hakupäivä 26.3.2011 http://www.ketteratkaytannot.fi/fi-FI/Kaytannot/Refaktorointi/

Szalvay, V. 2007. Glossary of Scrum Terms. Hakupäivä 30.3.2011 http://www.scrumalliance.org/articles/39-glossary-of-scrum-terms

Tieturi Oy. 2010. Certified ScrumMaster version 2.4. (Ei julkaisupaikkaa eikä julkaisijaa).

Virtanen, P. Johtava konsultti, Tieturi Oy. 2010. Luento 17.6.2010. Oulu.

Wells, D. 2009a. Extreme Programming: A gentle introduction. Hakupäivä 14.3.2011 http://www.extremeprogramming.org/

Wells, D. 2009b. The Values of Extreme Programming. Hakupäivä 14.3.2011 http://www.extremeprogramming.org/values.html

Yrittäjä, toimeksiantajayritys. 2009. Haastattelu 4.12.2009. Tekijän hallussa.

ESIMERKKI TAULUKKOMUOTOISESTA TUOTTEEN TYÖLISTASTA

LIITE 1

					New Estimates of Effort Remaining as of Sprint…					
Item	**Details (wiki URL)**	**Priority**	**Estimate of Value**	**Initial Estimate of Effort**	**1**	**2**	**3**	**4**	**5**	**6**
As a buyer, I want to place a book in a shopping cart (see UI sketches on wiki page)	…	1	7	5						
As a buyer, I wan to remove a book in a shopping cart	…	2	6	2						
Improve transaction processing performance (see target performance metrics on wiki)	…	3	6	13						
Investigate solutions for speeding up credit card validation (see target performance metrics on wiki)	…	4	6	20						
Upgrade all servers to Apache 2.2.3	…	5	5	13						
Diagnose and fix the order processing script errors	…	6	2	3						
As a shopper, I want to create and save a wish list	…	7	7	40						
As a shopper, I wan to add or delete items on my wish list	…	8	4	20						

KUVIO 11. Tuotteen työlista (Deemer ym. 2010, 8)

ESIMERKKI TAULUKKOMUOTOISESTA SPRINTIN TEHTÄVÄLISTASTA

LIITE 2

				New Estimates of Effort Remaining as of Day …					
Product Backlog Item	**Sprint Task**	**Volunteer**	**Initial Estimate of Effort**	**1**	**2**	**3**	**4**	**5**	**6**
As a buyer, I want to place a book in a shopping cart	modify database	Sanjay	5	4	3	0	0	0	
	create webpage (UI)	Jing	3	3	3	2	0	0	
	create webpage (Javascript logic)	Tracy & Sam	2	2	2	2	1	0	
	write automated acceptance tests	Sarah	5	5	5	5	5	0	
	update buyer help webpage	Sanjay & Jing	3	3	3	3	3	0	
	…	…	…						
Improce transaction processing performance	merge DCP code and complete layer-level tests		5	5	5	5	5	5	
	complete machine order for pRank		3	3	3	3	3	3	
	change DCP and reader to use pRank http API		5	5	5	5	5	5	
	…	…	…						
	…	…	…						
	Total (person hours)		50	49	48	44	43	34	

KUVIO 12. Sprintin tehtävälista (Deemer ym. 2010, 13)

SCRUM-PROSESSIN TOIMIVUUTEEN LIITTYVIÄ MITTAREITA LIITE 3

Taulukko 2. Hartmannin ja Stallingsin kehittämiä Scrum-prosessin toimivuuteen liittyviä mittareita (Scrum Process Metrics) (Leffingwell.& Smits 2005, 26)

Project:		
Scoring (0–5)	**Sprint 1**	**Sprint 2**
1. Product Owner		
Product Backlog developed, owned and managed by Product Owner		
Product Owner prcess is flexible; collaboration with team is ongoing		
Product Owner and stake-holder participation at Sprint Review		
Product Owner manages project by value		
Total "Product Owner" Score		
2. Planning		
Product Backlog is descriptive, prioritized and has effective estimates		
Team develops and manages Sprint Backlog		
Team involves stakeholders and dependencies in effective manner		
Project progress can be tracked by backlog BurnDown and value burnup		
Sustainable pace		
Total "Planning" Score		
3. Schedule		
Sprint Planning regular, on time, fully attended		
Sprint Review regular, on time, fully attended		
Daily Scrum occurs on time, is fully attended		
Team meets its commitments to Sprint		
Total "Schedule" Score		
4. Process		
Team self-polices and reinforces use of process and rules		
Organization is able to comply with Scrum rules		
ScrumMaster is effective in getting process followed		
Team is self-managing		
Surprises don't occur		
Team is cross-functional		
Team and Product Owner collaborate and work closely together		

Team works to improve itself and its processes		
Team adequately manages dependencies		
Total "Process" Score		
5. Team		
Team members are dedicated and honor commitments		
Team effectively acts upon indicators in Sprint Burndown		
Communications between team members is effective		
Team effectively manages conflict within team		
Team improves internal development processes		
Total "Team" Score		
6. Reporting		
Product Backlog is effectively maintained and communicated		
Sprint Backlog is effectively maintained and communicated		
Project and Sprint reporting are effective and understood		
Value is used to manage Product Backlog		
Total "Reporting" Score		
7. Engineering Practices/Infrastructure		
At least daily build		
Common source code library		
Metric for quality of increment		
Increment metric met		
Test driven development practices		
Refactoring practices		
Design review practices		
Code review practices		
Code standards		
Automated unit test harness		
Automated acceptance test harness		
Total "Engineering" Score		

ESIMERKKEJÄ SCRUMIN KÄYTTÖÖNOTOSSA HAVAITUISTA ONGELMISTA

LIITE 4

Taulukko 3. Esimerkkejä Scrumin käyttöönotossa havaituista ongelmista (Leffingwell.& Smits 2005, 16)

	Impact	Cost	Owner
Scrum Process			
People arrive late to daily Scrum and to not support basic discipline			
Scrum meeting take too long – team is bored and considers the time unproductive			
ScrumMaster dictates design decisions or micromanages			
Teams are too large for effective daily Scrum and Sprint planning			
Teams do not report task remaining time for BurnDown analysis			
People Practices			
Individuals interrupted and tasked to work outside the Sprint			
Teams isolated in cubicle and not in open Scrum area			
Team members not accountable for personal Sprint commitments			
Individuals are multiplexed across too many projects and teams			
Product Engineering Practices			
Cross-functional resources for definition, implementation and test are not present on the team			
Sprints do not fully implement and test potentially deployable increments of customer valued features			
Product owner not easily available/not integral to team			
System integration is not forced at each Sprint			
Product owner won't split up massive product backlog items to fit within a Sprint			
Teams have ineffective resources for automating builds and integrations			
Features loaded into Sprint after Sprint begins			
Organizational Issues			
Software process police regulate to ineffective processes			
Management assumes fixed price, fixed time, fixed functionality delivery posture			

Software Test and/or System QA is separate organization and is not integrated with team			
Organization rewards individual, rather than team behavior			
Existing rules or software capitalization demand adherence to document-driven, waterfall approaches			
Teams not co-located to maximum extent feasible			
Teams cannot make small organizational, space and expense decisions needed to do their job			

Legend:

Impact of this impediment to the project (0–9, 0 low, 9 high)

Cost to resolve this impediment (0–9, 0 low, 9 high)

Owner: Point in an organization for resolution (C – CEO/CTO/COO/CFO, V – VP, D – Director, P – Product Management, E – Engineering Management, T – Team)

HAASTATTELUKYSYMYKSET LIITE 5

Toimeksiantajayrityksen edustajille esitetyt haastattelukysymykset. Kysymyksen perässä oleva (P) merkitsee, että asiaa on kysytty vain projektipäälliköltä, (O) että kysymys on esitetty vain ohjelmistosuunnittelijalle.

Nykyinen projekti- ja tiimityöskentely

- Kuinka monta ihmistä tyypillisessä tiimissä on?
- Kuinka monta projektia on käynnissä yhtä aikaa? (P)
- Työskenteleekö eri projekteissa eri henkilöitä vai työskenteleekö yksi henkilö monessa projektissa? (P)
- Millainen ammattillinen koulutus tiimin jäsenillä on? (P)
- Millainen ammatillinen koulutus sinulla on? (O)
- Löytyykö tiimin jäseniltä osaamista kaikista tarvittavista kehitystyön osa-alueista?
- Millaisiin asioihin projektinhallinnen kehittämisellä toivotaan parannusta? (P)
- Millaisiin konkreettisiin ongelmiin olet törmännyt projektien hallinnassa ja toteutuksessa?
- Kuinka pysyviä tiimien kokoonpanot ovat? (P)
    - Hyödynnetäänkö yrityksessä harjoittelijoita? (P)
- Minkä parissa työskentelet tällä hetkellä, mistä vastaat ja millaisia tehtäviä teet?
- Millainen käsitys sinulla on siitä, mitä muut tiimin jäsenet milloinkin tekevät?
- Miten helppo on jatkaa työtä, jonka joku toinen tiiminjäsen on aloittanut? (O)
- Miten projektissa tapahtuvat muutokset tulevat esille?
    - Miten niihin suhtaudutaan?
- Onko tiimille asetettu yhteistä tavoitetta?
    - Mikä se on tämän hetkisessä projektissa?
- Antavatko tiiminjäsenet toisilleen palautetta?
    - Annatko sinä palautetta?
    - Miten itse koet palautteen?
- Kuinka sitoutuneita työntekijät ovat projektin toiminnan kehittämiseen? (P)
- Jakavatko tiimin jäsenet tärkeitä projektiin liittyviä tietoja toistensa kanssa?
    - Oletko huomannut joidenkin asioiden tulevan toisille yllätyksenä?
- Onko tiimin jäsenten välinen kommunikointi mielestäsi riittävää?

- Toimiiko tiimi mielestäsi tehokkaasti?
    - Missä olisi parantamisen varaa?

Scrumiin liittyvät kysymykset

- Miksi juuri Scrum kiinnostaa yritystä? (P)
    - Mitä hyötyä Scrumin odotetaan tuovan? (P)
- Onko yrityksessä jo pohdittu/päätetty, kuka ottaa hoitaakseen Scrum-mestarin tehtävät? (P)
    - Onko huomioitu tämän tehtävän hoidon vaatima aika? (P)
- Onko tiimin jäsenillä aikaisempaa Scrum-koulutusta? (P)
- Onko sinulla aikaisempaa Scrum-koulutusta (O)
    - Koetko tarvitsevasi lisätietoa, mikäli Scrumin käytäntöjä päätetään ottaa käyttöön?
- Onko asiakkailla kykyä ja halukkuutta sitoutua Scrumin käyttöön? (P)
- Onko yrityksessä varauduttu kustannuksiin, joita Scrumin käyttöönotto tuo? Esimerkiksi tuottavuuden hetkellinen lasku, koulutuspäivät, ulkopuolisen asiantuntijan käyttäminen. (P)
- Mitä Scrumin osa-alueita yrityksessä on jo otettu käyttöön?
- Miten tässä osittaisessa käyttöönotossa on mielestäsi onnistuttu?
- Onko käyttöönotettuja käytäntöjä noudatettu?
    - Millaisia ongelmia Scrumin käytössä on esiintynyt?
    - Mikä on toiminut hyvin? Mistä on ollut hyötyä?

- Onko jotain muuta projektikäytäntöihin liittyvää, josta ei ole osattu kysyä ja josta on sanottavaa?